Final Report

Contract /Grant Number : N00014-93-1200

Agent: Office of Naval Research

Contract Title: Outdoor Landmark Recognition Using Hybrid Fractal Vision System and Neural Networks

Contractor/Organization: North Carolina State University

Principal Investigator: Dr. Ren C.Luo
Department of Electrical and
Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911
Tel. 919-515-5199; Fax. 919-515-5523
e-mail: luo@eos.ncsu.edu

Actual Start Date: October 1,1993

Expected End Date: December 31 1996

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Feburary 24,1997	Final Report Oct. 1993 - Dec. 1996

4. TITLE AND SUBTITLE OF REPORT
Outdoor Landmark Recognition Using Hybrid Fractal Vision System and Neural Networks

5. FUNDING NUMBERS
N00014-93-1200

6. AUTHOR(S)
Dr. Ren C. Luo

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911

8. PERFORMING ORGANIZATION REPORT NUMBER:

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Ballston Tower 1
800 N. Quincy Street
Arlington, VA 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER:

11. SUPPLEMENTARY NOTES:

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release, distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The objective of this research project is to develop a hybrid vision system to isolate and recognize landmarks in outdoor natural scenes. We have used two-steps approach to solve this problem. The first step is to isolate the landmarks based on image segmentation using fractal models for objects in the images, and the second step is to recognize the isolated landmarks using neural networks. An Incremental Fractional Brownian Motion(IFBM) model for segmenting and isolating the outdoor natural images have been developed and successfully tested. Two neural networks systems namely, a reconfigurable multilayered feed forward neural network and self organizing neural netwok, with adaptive learning capabilities have been designed and trained on a variety of different outdoor traffic signs. The networks are robust and efficient and can recognize signs successfully with vigorous test.

We have implemented this hybrid system onto an autonomous mobile robot to achieve vision based navigation with natural landmark recognition. The system has been successfully tested in the lab environment laid-out with a variety of traffic signs. The robot can navigate through scattered environment with random obstacles and traffic signs. The robot has won twice championship on international autonomous mobile robot research and application competition in 1993 and 1995, respectively. The contenders consists of universities, government laboratories and companies from both US and outside of US. The competition was sponsored by AAAI (American Association of Artificial Intelligence).

The results of this research can be extended to a variety of both military and commercial applications such as military surveillance,autonomous vehicle navigation,automatic target recognition, indoor autonomous service robot navigation, object inspection and recognition etc.

14. SUBJECT TERMS
Fractal Vision, Neural Network, Fractal Model

15. NUMBER OF PAGES:

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT:	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

ABSTRACT

The objective of this research project is to develop a hybrid vision system to isolate and recognize landmarks in outdoor natural scenes. We have used two-steps approach to solve this problem. The first step is to isolate the landmarks based on image segmentation using fractal models for objects in the images, and the second step is to recognize the isolated landmarks using neural networks.

An Incremental Fractional Brownian Motion(IFBM) model for segmenting and isolating the outdoor natural images have been developed and successfully tested. Two neural networks systems namely, a reconfigurable multilayered feed forward neural network and self organizing neural netwok, with adaptive learning capabilities have been designed and trained on a variety of different outdoor traffic signs. The networks are robust and efficient and can recognize signs successfully with vigorous test.
We have implemented this hybrid system onto an autonomous mobile robot to achieve vision based navigation with natural landmark recognition. The system has been successfully tested in the lab environment laid-out with a variety of traffic signs. The robot can navigate through scattered environment with random obstacles and traffic signs. The robot has won twice championship on international autonomous mobile robot research and application competition in 1993 and 1995, respectively. The contenders consists of universities, government laboratories and companies from both US and outside of US. The competition was sponsored by AAAI (American Association of Artificial Intelligence).

The results of this research can be extended to a variety of both military and commercial applications such as military surveillance,autonomous vehicle navigation,automatic target recognition, indoor autonomous service robot navigation, object inspection and recognition etc.

Number of ONR supported:

Papers published in Refereed Journals: 6
Papers accepted for publication in refereed Journals: 22
Papers or reports in non-referred Journals: 0
Books or book chapters published: 1

Number of Presentations
Invited: 3
Contributed: 19

Trainee Data:

No. of graduate students: Total: 3 Female: 0 Male: 3 Minority: 0 Non-US: 2
No. of postdoctorals: 0
no. of Undergraduates: 0

Number, cost and description of equipment items costing more than $1,000 that were purchased on your ONR grant------- N/A

Awards/Honors to PI and/or to member of PI's research group:

We have incorporated our vision research results to our MARGE Autonomous Mobile Robot. One test of our system's ability to handle a real task was observed why MARGE was entered in the 1993 Autonomous Robot Competition sponsored by the American Association of Artificial Intelligence. MARGE won first prize in the event called "Office Rearrangement". This event required the robot to search for specially marked boxes in the competition arena, and move them into a pattern at a specified location. We have used MARGE as a testbed for demonstrating our vision system together with Neural Network algorithms developed under this research. We have greatly improved system and research findings since 1993. We also entered the 1995 AAAI contest which was held on August 20-25, 1995 in Montreal, Canada. We won the first place again for the event called "Robot Clean Up the Offices". The contenders consisted of universities, government labs, companies from both US and outside of US.

The PI, Prof. Ren C. Luo has received ALCOA Foundation Distinguished Engineering Research Award from North Carolina State University in May 1996.

Possible Transitions:

We feel that the following are relevant transitions of our research effort.

Military applications:
- Military surveillance
- Automatic target recognition
- Autonomous vehicle navigation

Commercial applications:
-Material handling for factory automation
-Object inspection and recognition
-Date compression/decompression
-Telecommunication/teleconference
-Indoor autonomous service robot navigation

PUBLICATIONS

Book

"Multisensor Integration and Fusion in Intelligent Machines and Systems," Ablex Publisher, Norwood, NJ, 1994.

Refereed Journal Publications

1. R. LeGrand and R. C. Luo, "Lola: Object Manipulation in an Unstructured Environment", AI Magazines, Vol. 6, No. 1, 1996.
2. R. Gutierez and R. C. Luo, "Lola: Probabilistic Mobile Robot Navigation for Topological Maps", AI Magazines, Vol. 6. No. 1, 1996.
3. S. Goodridge, R. C. Luo, and M. G. Kay, "Hirachical Fuzzy Behavior Fusion and Control Using Multiple Sensors", in IEEE Transactions on Industrial Electronics Vol. 43 No. 3, 1996.
4. H. Potlapalli and R. C. Luo, "Projection Learning for Self-Organizing Neural Networks", IEEE Transactions on Electronics, Vol. 43, No. 2, 1996.
5. J. A.. Janet, R. Gutirez, and R. C. Luo, "Autonomous Mobile Robot Self Localization Using Kohonen and Feature Based Mobile Robot", Journal of Robotics Systems, Vol. 14, No.4, 1997.
6. J. A. Janet, R. C. Luo, and M. G. Kay, "Autonomous Mobile Robot Global Motion Planning and Geometric Beacon Collection Using Traversability Vectors" IEEE Transaction on Robotics and Automation, Vol. 13, No. 1, 1997.

Refereed Conference Publications

1. R. C. Luo, H. Potlapalli, and D. W. Hislop, "Outdoor Landmark Recognition Using Factal Based Vision and Neural Networks", International Conference on Robots and Systems, Yokohama. Japan, 1993.
2. R. C. Luo, C. M. Aras, J. Janet, and M. G. Kay, "Sonar Windows and Geometrically Represented Objects for Robotics Systems", Yokohama, Japan, July 1993.
3. R. C. Luo, H. Potlapalli, and D. W. Hislop, "Mobile Robot Navigation Using Fractals and Neural Network", in Proceeding of the First World Congress on Intelligent Control and Intelligent Automation, Beijing, China, August 1993.
4. S. G. Goodridge and R. C. Luo, "Fuzzy Behavior Fusion for Autonomous Mobile Robot Control", IEEE International Conference on Fuzzy Theory and Technology, Durham, NC, Oct. 1993.
5. R. C. Luo, H. Potlapalli, and D. W. Hislop, "Defocusing Blur Restoration in Natural Scene Images for Fractal Analysis", IEEE Intl. Conf. on Industrial Electronics, Hawaii, Novmber 1993.
6. J. A. Janet, S. G. Goodridge, and R. C. Luo, "Dynamic Motion Control of Sensor Based Autonomous Mobile Robot", J. A. Janet, S. G. Goodridge, and R. C. Luo, invited paper in International Conference on Mechatronics and Robotics. Aachen, Germany, April 1994.

7. R. C. Luo, H. Potlapalli, and D. W. Hislop, "Landmark Recognition for Mobile Robots in Dynamic Environment Using Self Organizing Neural Networks", IEEE Robotics and Automation Conference, San Diego, CA, Jyly 1994.
8. S. G. Goodridge, and R. C. Luo, "Fuzzy Behavior Fusion for Reactive Control of an Autonomous Mobile Robot", IEEE Robotics and Automation Conference, San Diego, CA, May 1994.
9. H. Poltapalli and R. C. Luo, "Landmark Recognition Using Projection Learning for Mobile Robot Navigation", Invited Paper, IEEE World Congress on Computational Intelligent, Orlando, FL, July 1994.
10. J. A. Janet and R. C. Luo, "Autonomous Mobile Robot Navigation Using Traversibility Vectors", IEEE International Conference on Industrial Electronics, Bologna, Italy, Sept. 1994.
11. J. A. Janet and R. C. Luo, "MARGE: an Autonomous Mobile Robot Using Multisensors and Fuzzy Control", 1994 International Conference on Intelligent Robots and Systems, Munich, Germany, Sept. 1994.
12. J. A. Janet, M. G. Kay, and R. C. Luo, "The Essential Visibility Graph: An Approach to Global Motion Planning for an Autonomous Mobile Robot", 1995 IEEE International Conf. on Robotics and Automation, Nagoya, Japan, May 1995.
13. A. Janet, R. Gutierrez-Osuna, T. A. Chase, and R. C. Luo, "Global Self-Localization for Autonomous Mobile Robots Using Self-Organizing Kohonen Networks", 1995 IEEE/RSJ International Conf. on Intelligent Robot and Systems, Pittsburg. PA, August 1995.
14. J. A. Janet, R. Gutierrz-Osuna, and T. A. Chase, "Global Self- Localization for Autonomous Mobile Robots Using Region-and Feature- Based Neural Network", 1995 IEEE International Conf. on Industrial Electronics, Orlando, Fl, Nov. 1995.
15. J. A. Janet, T. A. Chase, and R. C. Luo, "Pattern Analysis for Autonomous Vehicles with the Region-and Feature-based Neural Network, Global Self-Localization and Traffic Sign Recognition", IEEE International Conf. on Robotics and Automation, Minneapolis, Mn, May 1996.
16. R. LeGrand and R. C. Luo, "Position Estimation of Selected Targets", IEEE International Conf. on Robotics and Automation, Minneapolis, Mn, May 1996.
17. J. A. Janet, R. Gutierrz-Osuna, and R. C. Luo, "Autonomous Mobile Robot Sonar Range Prediction and Credence with Sonar Windows", IEEE International Conf. on Robotics and Automation, Minneapolis, Mn, May 1996.
18. M. S. Azam and R. C. Luo, "A. Hybrid Vision System for Meaningful Landmark Isolation and Recognition," IEEE International Conf. on Industrial Electronics, Taipei, Taiwan, R. O. C.., August 1996.
19. S. G. Goodridge and R. C. Luo, "Multisensor Based Fuzzy Behavior Fusion for Real Time Control of Automatons Mobile Robot", (Invited Paper) Proceedings of Chinese Society of Mechanical Engineering, Taipei, Taiwan, R. O. C., August 1996.

20. J. A. Janet, D. Schudel, and R. C. Luo, "Global Self Localization for Actual Robots: Generating and Sharing Topological Knowledge Using the Region Feature Neural Network", IEEE International Conf. on Multisensor Fusion and Integration for Intelligent Systems, Washington. D. C., August 1996.
21. T. Hamada and R.C.Luo," Intelligent Mobile Robot Capable of Recognizing Gestues and Taking Actions ", accepted in International conf. on Intelligent Control and Automation Xi'an, China, June 1997
22. R.C.Luo and T.M. Chen," Remote Supervisory Control of Intelligent Autonomous Mobile Robot via World Wide Web" accepted in IEEE International Symposium on Industrial Electronics " Minho, Portuge, July 1997.

Fractal Based Classification of Natural Textures

Harsh Potlapalli*, Ren C. Luo*, David W. Hislop+
*Center for Robotics and Intelligent Machines,
North Carolina State University,
Raleigh, NC 27695-7911
+US Army Research Office
Research Triangle Park, NC 27709-2211

Abstract

Texture classification is an important first step in image segmentation and image recognition. The classification algorithm must be able to overcome distortions such as scale, aspect and rotation changes in the input texture. In this paper, a new fractal model for texture classification is presented. The model is based on Fractional Brownian Motion. It is shown that this model is invariant to changes in incident light intensity as well. The isotropic nature of Brownian Motion is particulary useful for outdoor applications where viewing directions may change. Classification results of this model are presented and compared to other texture measurement models.

1 Introduction

Texture can be defined as a coarseness, or, roughness measure of a surface. Texture measurements can be very useful for surface classification experiments. A unique texture measurement that can separate a surface from other similar surfaces is very critical to object recognition and image segmentation applications. In the case of natural scenes, texture classification is useful to differentiate between the roads, vegetation, sky as well as the vehicles, traffic signs, etc.

With regard to texture analysis and object classification, there have been several statistical approaches to the measurement and characterization of image texture such as textural edgeness [1], spatial gray level co-occurrence probabilities [2], etc. Fine textures tend to have high spatial frequencies, while coarse textures tend to have low spatial frequencies. Kashyap and Eom studied the correlation sequences over varying distances to obtain a texture feature vector that could be used for classification [3].

Haralick, et al., used a vector formed by angular second moment, the contrast and the correlation within the region [4]. The "texture energy" in a window can be estimated by applying a set of convolving masks on the image [5].

In some cases, the region was treated as a sample of a random process. It was assumed that the uniform regions were created by some pre-processing. Then the classification problem was reduced to identifying

the parameters of the random process. Kashyap and Khotanzad modeled the texture as a a realization of a symmetric autoregressive random field. The coefficients of this random process were estimated by the least squares method [6].

All the methods described above work well for specific applications that they were designed for. However, these methods have poor generalizing capabilities. Feature vectors are not reliable in cases where the lighting conditions are varying. Spatial statistics matrices are not invariant to rotations. They also require large amounts of computation. This is also true for the texture energy approaches. In unstructured environments where the object parameters, such as size and orientation, are changing, these methods do not give consistent results.

Fractals offer an alternative to these approaches. Fractal surfaces can be considered to be those surfaces whose topological dimension takes on non-integer values. Most natural surfaces, such as coastlines, brick, skin, rocks, can be modeled as fractal surfaces. Then the classification problem is reduced to estimating this fractal dimension. Since the fractal feature is an inherent property of the region/surface/object, it can be a more reliable feature.

2 Fractal Model of Textures

The theory of fractals has been used by a number of researchers in recent times to classify texture of natural surfaces as well as segment images of natural scenes. In case of real world textures, if some measure of the surface, such as area or length, is given by M when a measuring unit of size λ is used then

$$M = n\lambda^D \tag{1}$$

where D is a measure of the fractal dimension. D is a measure of the roughness of the surface. For example, the D value for geometric planes (which are very smooth) is lower than the D value for a sample taken from rock surfaces.

Mandelbrot and Van Ness developed a Fractional Brownian Motion (FBM) model to describe natural textures. [7]. Pentland derived several useful properties of the FBM in [8]. He showed that if a surface is fractal in nature the intensity image of the fractal surface was also fractal. He also proved the converse, i.e., that is if the image of a surface is fractal then the surface must be fractal. Another property he derived showed that the fractal dimension was invariant under linear transformations. These properties are very important to the applicability of the fractal model to the measurement of natural textures since these guarantee that the texture measured in the image is directly related to the actual texture, or roughness, of the surface. Pentland used this model to segment aerial images. In this method the image into $N \times N$ windows and computed the Fourier transform for each window. A linear regression was then performed on the pixels to estimate D.

Chen, et al., use high order statistical moments (greater than 2), to measure the fractal dimension [9]. The weighted mean pixel difference for every distance combination is computed within each $N \times N$ window. Next, linear regression is applied to this data to estimate the fractal dimension. Keller, et al., [10] used the FBM model to recover characteristics of natural scene from the silhouettes using a least-square linear fit in

estimating the fractal dimension.

Peleg used the definition of the fractal surface as a measure of surface "length" to estimate the surface roughness [11]. In this model a covering blanket that would just match the surface is iteratively computed. For each blanket, the volume enclosed is computed from which the area and thickness can be computed. To obtain a better estimate, Peleg suggests that a linear fit be performed between the the blanket thickness and the area enclosed. The slope of the best linear fit is the fractal dimension. Hentschel and Procaccia have extended this model by taking moments of the mean number of pixels of the surface that could be contained in a cube of side, b, and the total number of such cubes that would be needed to cover the entire surface [12]. Lundahl developed maximum likelihood estimator of the fractal dimension of bone texture using the properties of the increments of the FBM [13].

Some of the methods described above are very computation intensive. For example, Pentland's model requires the computation of the Fourier transform as well as a linear regression. Peleg's model requires an iterative volume computation approach. Linear regression is also required in Chen's model. In this paper we present an new approach to the measurement of the fractal dimension based on the increments of the FBM. By choosing an appropriate increment we derive a simple relationship between the image statistics and the fractal dimension. We present some useful properties that follow from this relationship. We also analyze the computational issues related to the measurements. Finally we will present the texture measurement results that are obtained with this model. We will also compare our results with those obtained by other models to show the accuracy of our model.

3 Definitions

First, we define the terms fractal dimension and fractal surface.

Consider a coordinate block C in R^n of the form $C = [(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)]$ where for all i, $b_i > a_i$. In other words, the (a_i, b_i) is a directed vector from a to b. Define the volume of C as $V(C) = (b_1 - a_1)(b_2 - a_2)...(b_n - a_n)$.

Definition 1 *If $E \subset R^n$, then the Lebesgue Measure of E is given by*

$$L^n(E) = min \sum_{i=} V(C_i) \tag{2}$$

where the min is taken over all coverings of E by a sequence C_i of blocks.

It is easy to prove that L^n is a metric. Also, from Falconer [14] we have a simple relationship between the Lebesgue measure and the Hausdorff measure.

Theorem 1 *The Fractal dimension of an m-dimensional smooth, continously differentiable manifold,E, is m.*

Proof: Since E is continously differentiable we can find the maxima and minima along each axis. We can also draw tangents to these points parallel to the corresponding axis. Let the tangent parallel to the i^{th} axis be the vector (a_i, b_i).

Next, we construct an enclosing box, C, made up of the set of tangents such that

$$C = [(a_1, b_1), (a_2, b_2), ..., (a_m, b_m)].$$

Since the surface exists in only m dimensions, only m such tangent pairs can be drawn. Also, the smallest coordinate block that will completely cover the surface is one that is made up of the tangents to the maxima and minima since this line is the closest to the surface without intersecting the surface. Therefore, the Lebesgue measure is given by

$$L^m = \sum_i V(C_i). \tag{3}$$

The Lebesgue measure is not defined for any $n > m$ since no tangents can be drawn along any of the $i_k, k = m + 1, ..., n$ axes. Also, the Lebesgue measure is incomplete for any $n < m$. Thus, the Lebesgue measure is defined only for $n = m$. Since the Hausdorff measure is directly related to the Lebesgue measure, the Hausdorff measure is finite and measurable only for $p = m$. Therefore, from the definition of the Fractal dimension, for a smooth continously differentiable m-dimensional manifold the Fractal dimension is m.

■

Definition 2 *A fractal surface is one that is continuous at every point but differentiable at no point.*

Theorem 2 *The Fractal dimension, p, of a fractal surface in R^n is greater than its topological dimension, m.*

Proof: This follows from Definition 2. Since the surface is not differentiable at any point no bounding tangents can be drawn at least within R^m. Also, we can always construct a bounding box in the next higher dimension. But this will not be minimum cover as required by the definition of the Lebesgue measure. Therefore, the Hausdorff measure will be finite for some p between m and $m + 1$.

■

Based on these definitions we can construct an estimator of the the fractal dimension. We will model the fractal surface as a Brownian Motion and develop the estimator from the statistics of the Brownian Motion.

4 Incremental Fractional Brownian Motion

Brownian Motion describes the path of a microscopic particle suspended in a liquid. Due to the atomic size of the particle, collisions with the molecules of the liquid cause frequent direction changes in the particle. All directions of motion (due to a collision) are equiprobable. The steps of the motion of the particle between collisions are identically distributed, independant and stationary.

Definition 3 *A Fractional Brownian Motion of index-H ($0 < H < 1$) is defined to be a random process X:[0,∞] → R on some probability space such that*

1. *X(t) is continous and X(0) = 0.*
2. *for any $t \geq 0$ and $\tau > 0$, the increments $X(t+\tau) - X(t)$ has normal distribution with mean, $\mu = 0$ and variance $\sigma^2 = \tau^{2H}$, that is,*

$$P[X(t+\tau) - X(t) \leq x] = \frac{1}{\sqrt{2\pi}\,\tau^{-H}} \int_{-\infty}^{x} exp[\frac{-u^2}{2\tau^{2H}}]du \tag{4}$$

Theorem 3 *An index-H FBM, X:[0,1] → R has a graph with Fractal dimension 2-H.*

The outline for the proof of this theorem along with other related analysis on Fractional Brownian Motion has been introduced by Falconer in [14]. We have developed extensions of these proofs in [15].

In this paper, we show a new fractal dimension estimator. We define *Incremental Fractional Brownian Motion* (IFBM) for discrete time as follows. If B(n) is Brownian motion then the IFBM of order m, $I_m(n)$, is given by

$$I_m(n) = B(n) - B(n-m) \tag{5}$$

Lundahl, et al, have evaluated the fractal dimension by applying maximum likelihood estimators to the IFBM, [13]. Here, we present another solution using second order moments.

Let the variance of the IFBM, var$[I_m(n)]$ be given by σ^2. From the properties of the FBM we have

$$E[B(n) - B(n-m)]^2 = \beta\,[(n) - (n-m)]^{2H} \tag{6}$$

But, from the definition of the IFBM we have,

$$E[B(n) - B(n-m)^2] = E[I_m(n)^2] \tag{7}$$

That is, the variance of the IFBM can be expressed as

$$\beta[(n) - (n-m)]^{2H} = \sigma^2 \tag{8}$$

If we expand the inner terms, the n cancel out and for $m = 1$ we can write equation 8 as

$$\beta = \sigma^2 \; for \; m \; = \; 1 \tag{9}$$

Now, we will begin to develop a relationship between the fractal dimension and the image statistics. The autocorrelation of the IFBM is given by

$$R_{I_m I_m}(n+k, n) = E[I_m(n+k) I_m(n)] \tag{10}$$

Since the IFBM is a stationary process, the autocorrelation must be a function of the time shift or in the other words the difference, (n+k) - (n) = k. Also, substituting the FBM terms for $I_m(n)$ we have

$$R_{I_m I_m}(k) = E[\{B(n+k) - B(n+k-m)\}\{B(n) - B(n-m)\}] \tag{11}$$

By expanding the terms on the right hand side and by evaluating the expectation of each product term separately we obtain the following simplified expression for the autocorrelation.

$$\begin{aligned} R_{I_m I_m}(k) &= -\frac{1}{2}E[\{B(n+k) - B(n)\}^2] + \frac{1}{2}E[\{B(n+k) - B(n-m)\}^2] \\ &\quad -\frac{1}{2}E[\{B(n+k-m) - B(n)\}^2] \\ &\quad -\frac{1}{2}E[\{B(n+k-m) - B(n-m)\}^2] \end{aligned} \tag{12}$$

From the properties of the FBM and the result obtained in equation (8) each squared difference can be expressed as a product of the variance of the IFBM and the time shift. Taking the common factor σ^2 outside the sum, we have

$$R_{I_m I_m}(k) = \frac{\sigma^2}{2}\left\{(k+m)^{2H} - 2k^{2H} + (k-m)^{2H}\right\} \tag{13}$$

This result relates the correlation distance with the fractal dimension. The correlation and the variance are easily computable from the image data. We can simplify (13) by setting $k = m$. Then we have:

$$R_{I_m I_m}(m) = \frac{\sigma^2}{2}\left\{(2m)^{2H} - 2m^{2H}\right\} \tag{14}$$

This equation is still quite non linear. Also, the autocorrelation is low for large values of k (or, m). Hence, if choose $m = 1$ the autocorrelation reduces to

$$R_{I_m I_m}(1) = \frac{\sigma^2}{2}\left\{2^{2H} - 2\right\} \tag{15}$$

Equation (15) can be simplified as follows:

$$\begin{aligned} R_{I_m I_m}(1) &= \frac{\sigma^2}{2}\left\{2(2^{2H-1} - 1)\right\} \\ &= \sigma^2\left\{2^{2H-1} - 1\right\} \\ \frac{R_{I_m I_m}(1)}{\sigma^2} + 1 &= 2^{2H-1} \end{aligned} \tag{16}$$

Taking natural logs of both sides, we have

$$\ln\left[\frac{R_{I_m I_m}(1)}{\sigma^2}+1\right] = (2H-1)\ln(2)$$
$$\frac{1}{2}\left[\frac{\ln\left[\frac{R_{I_m I_m}(1)}{\sigma^2}+1\right]}{\ln(2)}+1\right] = H \tag{17}$$

Thus, the fractal dimension is a simple log ratio of the variance of the surface and the autocorrelation at distance of 1. Given that H varies from $\frac{1}{2}$ and 1,[7], we can compute the bounds of the log ratio. We have:

$$\frac{1}{2} < \frac{1}{2}\left[\frac{\ln\left[\frac{R_{I_m I_m}(1)}{\sigma^2}+1\right]}{\ln(2)}+1\right] < 1 \tag{18}$$

which can be reduced to:

$$0 < \ln\left[\frac{R_{I_m I_m}(1)}{\sigma^2}+1\right] < \ln(2) \tag{19}$$

The log ratio is thus between 0 and ln(2) or about 0.69. Dropping the logs and using the Cauchy-Schwartz inequality, we have

$$0 < \left[\frac{R_{I_m I_m}(1)}{\sigma^2}+1\right] < 2$$

or, in other words,

$$-1 < \frac{R_{I_m I_m}(1)}{\sigma^2} < 1 \tag{20}$$

Thus the actual ratio has a wider range from -1 to 1. This ratio can be used to provide greater discrimination between the textures. Also, the ratio is computationally cheaper to evaluate than the log function.

One interesting observation that can be made from this ratio is regarding the smoothness (or roughness) of natural textures. For smooth surfaces, the correlation between adjacent pixels is high since the intensity is fairly uniform over the entire surface. In this case the ratio of autocorrelation to the variance will be high. The resulting fractal dimension (=3-H) will be a low number. For rough surfaces the correlation will be low and the H-ratio will be small number leading to a high fractal dimension. This is best illustrated with the graph in Figure 1 showing three different one-dimensional FBMs with varying roughness. As expected we see that the R_{II}/σ^2 is highest for the smoothest motion (low fractal dimensions) and lowest for the roughest motion (high fractal dimension). This is consistent with the results reported in the literature that rougher surfaces tend to have higher fractal dimensions than smoother surfaces.

4.1 Intensity Invariance

From (17) and (19) we can show that the IFBM model is stable under changes in light intensity. Let $\vec{L}$(t) be the intensity of light incident on the object surface point under current observation let $\vec{S}$ be the surface reflectance function at the point. Then the observed surface intensity $B(t)$ is given by

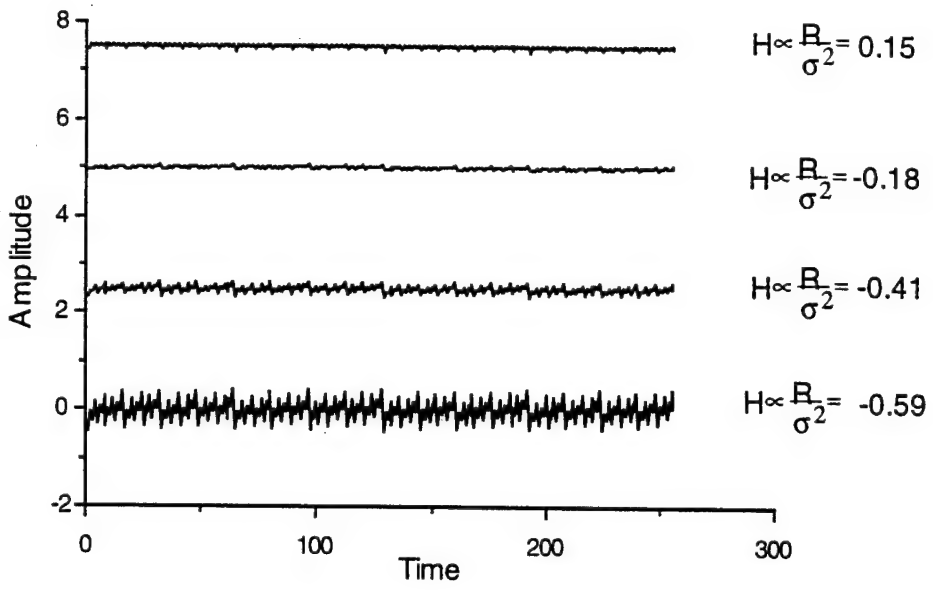


Figure 1: Relationship between roughness and corresponding correlation-variance ratio

$$B(t) = \vec{L}(t) \cdot \vec{S} \tag{21}$$

Now if the light intensity is changed to $\vec{L}'(t) = \alpha\ \vec{L}(t)$, then the new observed intensity B'(t) is given by

$$\begin{aligned} B'(t) &= \vec{L}'(t) \cdot \vec{S} \\ &= \alpha\ \vec{L}(t) \cdot \vec{S} \\ &= \alpha\ B(t) \end{aligned} \tag{22}$$

The variance can be expressed as $R_{I_m I_m}(0)$. Now the autocorrelation is the joint expectation $E[I(n)I(n+k)]$. If the light intensity changes by a factor of α then from the above we see that the observed values for the FBM model change from $B(n)$ to $\alpha B(n)$. The IFBM is then given by

$$\begin{aligned} I_m^{\alpha}(n) &= \alpha B(n) - \alpha B(n-m) \\ &= \alpha[B(n) - B(n-m)] \\ &= \alpha I_m(n) \end{aligned} \tag{23}$$

The autocorrelation is therefore

$$E[I\alpha_m(n)I\alpha_m(n+k)] = \alpha^2 E[I(n)I(n+k)]$$

Since this term appears both in the numerator as well as the denominator in (17) and (19) the α^2 terms cancel out leaving the same result as before. Thus the IFBM model is stable under changes in intensity.

4.2 Implementation

The computational complexity of the approach used to determine the fractal dimension is an important consideration especially in real time applications. For any region of size N x N using any criteria that depends on the parameters of the entire region, the lower bound on complexity is $O(N^2)$. For example, Fourier transforms are $O(N \log N)$ while the "blanket growing" method of Peleg is $O(k\ N^2)$. In the case of the current method, the computation of the variance is $O(N^2)$. The computation of the autocorrelation of unit distance is also $O(N^2)$. This computation is easily accomplished by considering only the 4 neighborhood of each pixel.

The single dimensional time sequence random processes described in the derivation of the IFBM model is a causal model. That is, to compute the IFBM at time t_1 we can consider only those values for B that exist for time $< t_1$. Images are non-causal; if the (row,column) co-ordinates are considered as equivalent to the time axis then we see that all values of the motion are available to us. In this case, we define the IFBM of order m at position (i, j) in the image as

$$I_m(i,j) = B(i,j) - \frac{B(i-1,j) + B(i,j-1) + B(i,j+1) + B(i+1,j)}{4} \tag{24}$$

That is, the IFBM at (i, j) is the average difference between the FBM at (i, j) and the 4-neighborhood of (i, j). Similarly, to compute the the 2-dimensional autocorrelation at unit distance (equation 15), we first multiply I(i,j) with each of its four neighbors and take the average. By repeating this over the entire region, the autocorrelation is determined. That is, the autocorrelation is given by

$$R_{II}(1) = \sum_i \sum_j I(i,j) \times \frac{I(i-1,j) + I(i,j-1) + I(i,j+1) + I(i+1,j)}{4}. \tag{25}$$

The fractal dimension estimate can be used for texture classification as described in the next section.

5 Results

In this section we describe the performance of the IFBM model. First we consider a set of standard natural textures. These are images of textures such as pebbles, paper, mica, fieldstone, beans and pellets [16].

5.1 Natural Textures

We considered 6 natural texture images to test the fractal dimension results obtained in equation 17. These images were obtained by anonymous ftp from ftp.teleos.com and were each 256x256 in size. Each image contained only one texture. The textures considered were: fieldstone, pebbles, pellets, mica, beans and paper as shown in Figure 2.

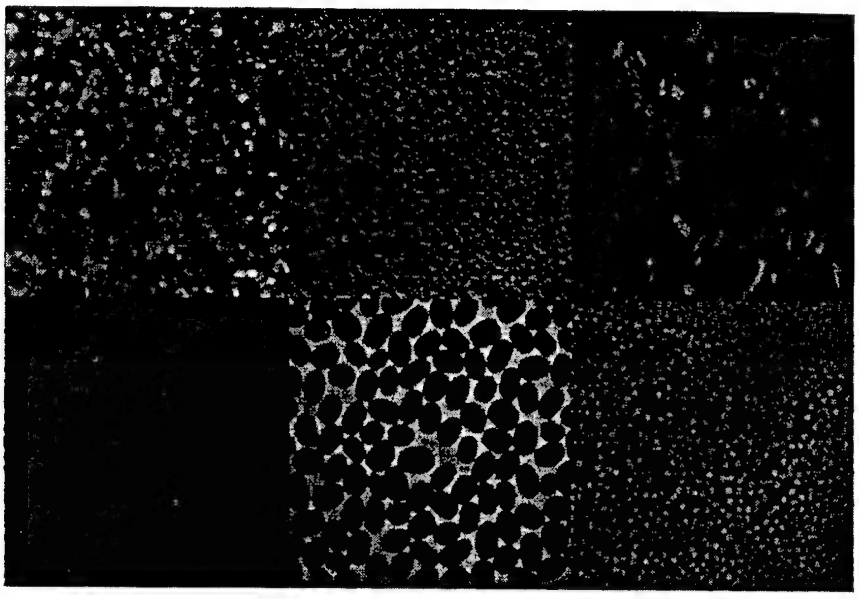


Figure 2: Textures used in classification experiments. Clockwise from top-left: Pebbles, Paper, Mica,Pellets,Beans,Fieldstone

To determine the performance of the IFBM classifier, the following experiment was performed. Our experiment consisted of the following steps. First, we chose sixteen 64×64 samples randomly from each texture. Next, from these sixteen we chose twelve samples to estimate the fractal dimension of the texture. That is, the twelve were used to "train" the classifier. The remaining four were used to test the classification performance. By choosing a pseudo-random number generator with a large period, we ensure that one set of sixteen samples is not selected more than once. This sequence was repeated 1000 times and the results were averaged out.

The performance of the fractal dimension as a classification tool is shown in Table 1. This table shows that the average accuracy of the IFBM classifier is 91.7%. This figure compares well with other fractal based texture classification results reported in the literature [17].

From this table we observe that some of textures such as Pebbles, Mica, Paper and Fieldstone have classified very well but others such as Beans and Pellets have not performed as well. To understand this difference we performed a goodness-of-fit test.

5.2 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov Test is a useful statistical tool to measure the goodness-of-fit between a predicted probabilistic model and the observed data [18]. This test is considered more reliable than a chi-squared test since it is independent of the number of intervals in the range of the data. To compute the difference we construct a cumulative distribution function of the observed data as well as the predicted model. The maximum difference between the two is compared with the difference allowed for a given level of significance. These

Texture	1	2	3	4	5	6	Accuracy
1. Pebbles	4						100
2. Paper		4					100
3. Mica			4				100
4. Fieldstone				4			100
5. Beans					3	1	75
6. Pellets					1	3	75

Table 1: Classification Results Using the IFBM Model

Texture	K-S Difference	Decision
1. Pebbles	0.014	Accept
2. Paper	0.019	Accept
3. Mica	0.016	Accept
4. Fieldstone	0.017	Accept
5. Beans	0.101	Reject
6. Pellets	0.128	Reject

Table 2: Kolmogorov-Smirnov Difference

can be taken from any handbook of statistics. For a 64×64 sized data set containing 4096 points the maximum difference allowed for a 0.05 level of significance (probability of rejecting predicted model when it is true) is 0.02.

Table 2 gives the maximum difference for the textures tested for the classification. We notice that the textures that gave poor classification also have poor fits between the data and the predicted model. Thus, these textures cannot be considered as fractal textures.

Figure 3 shows the cumulative distribution functions of the textures. Here we are checking if the image data fits the IFBM model (or, the underlying Gaussian distribution). In the case of the Beans texture, we notice that the image is not continuous in the sense that there are regions where there is no texture. We also notice this in the Pellets image which, too, has a poor fit between the observed data and the predicted model. It can be theorized that the IFBM model performs best in the case of textures that cover the entire span of the observation window and that the model will perform poorly in cases where there are gaps or holes in the texture. Also, in the case of regions that contain overlapping or multiple textures, the estimated fractal dimension will bear little relation to the fractal dimension of any of the textures in the region. This is in fact true of any region

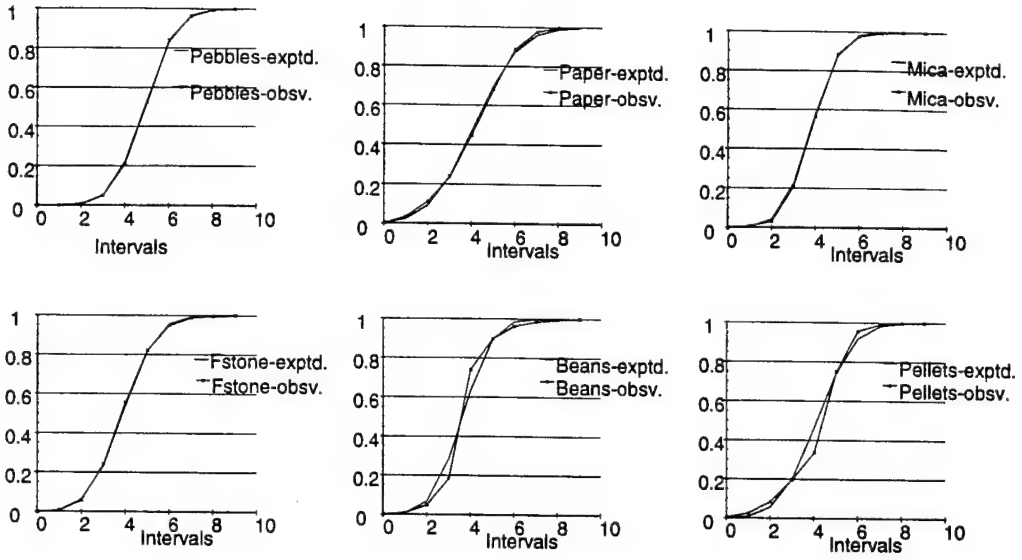


Figure 3: Cumulative Distribution Function of Texture and Predicted Model; X-axis is the interval, Y-axis is the CDF; Clockwise from top-left textures are: Pebbles, Paper, Mica, Pellets, Beans, Fieldstone

analysis method.

5.3 Comparisons with Other Models

We compared the performance of the IFBM model with other texture measurement methods. There are three principal ways texture is measured as seen earlier in this chapter: spatial statistics, local texture energy using convolving masks, and random field models. We choose the Spatial Gray Level Dependence Matrix method of Conners [19], the Local Texture Energy masks method by Laws [5] and the symmetric autoregressive model by Kashyap and Khotanzad [6] as representative of each type. The classification results in each case are presented below.

The Spatial Gray Level Dependence Method has an accuracy of 79%, the Local Texture Energy method has an accuracy of 70% and the random field method has an accuracy of 83%. The SGLDM method generates a 12 dimensional feature vector and it requires four large matrix to store the intensity variations in the four directions. The LTE method generates a 9 dimensional feature vector. Since these two methods require computations of the intensity variations in specific directions, these methods are very sensitive to changes in orientation. All three methods are based on direct computation of the intensity values; hence they are very sensitive to changes in incident intensity. The IFBM method presented in this dissertation does not have the limitations of direction or incident light. It also has a higher classification accuracy. Hence, the IFBM model is a superior texture measurment model.

We compared the IFBM model to two other fractal models: the Power Spectral Density (PSD) model

Texture	1	2	3	4	5	6	Accuracy
1. Pebbles	3	1					75
2. Paper	1	3					75
3. Mica			3	1			75
4. Fieldstone			1	3			75
5. Beans					4		100
6. Pellets			1			3	75

Table 3: Classification Results Using the SGLDM Model: 79% accuracy

Texture	1	2	3	4	5	6	Accuracy
1. Pebbles	3	1					75
2. Paper		4					100
3. Mica			3	1			75
4. Fieldstone			2	2			50
5. Beans			1		3		75
6. Pellets	1	1				2	50

Table 4: Classification Results Using the Local Texture Energy Model: 70% accuracy

Texture	1	2	3	4	5	6	Accuracy
1. Pebbles	4						100
2. Paper		4					100
3. Mica			4				100
4. Fieldstone			1	3			75
5. Beans				1	2	1	50
6. Pellets				1		3	75

Table 5: Classification Results Using the Random Fields Model: 84% accuracy

Texture	Pebbles	Paper	Mica	Fieldstone	Beans	Pellets	Accuracy
Pebbles	4						100
Paper		4					100
Mica			4				100
Fieldstone				4			100
Beans					4		100
Pellets					1	3	75

Table 6: Classification results using the PSD Fractal model; Average accuracy = 96%

Texture	Pebbles	Paper	Mica	Fieldstone	Beans	Pellets	Accuracy
Pebbles	4						100
Paper		4					100
Mica			3	1			75
Fieldstone				4			100
Beans					2	2	50
Pellets				1		3	75

Table 7: Classification results using the Box-Counting Fractal model; Average accuracy = 84%

(or, the Fourier model), [8] and the Box-Counting Model developed by Loh [20]. The PSD method has higher classification accuracy than the IFBM model (see Table 6. Recall, though, that the PSD is the Fourier transform of the autocorrelation function. Therefore, the PSD method requires additional computation compared to the IFBM model in which we can estimate the fractal dimension directly from the autocorrelation. The trade-off between the two methods is high computation requirements and high accuracy versus lower computation requirements and slightly lower accuracy. The box-counting method has lower accuracy (see Table 7) which is also borne out in experiments conducted in [10]. It had the same order of computational complexity as the IFBM model. Loh was able to improve the classification performance by incorporating additional fractal dimension estimators.

The fractal dimension estimate can be seen to be a useful first step in distinguishing between textures. The fractal estimate can be coupled with other knowledge based reasoning algorithms for image segmentation problems.

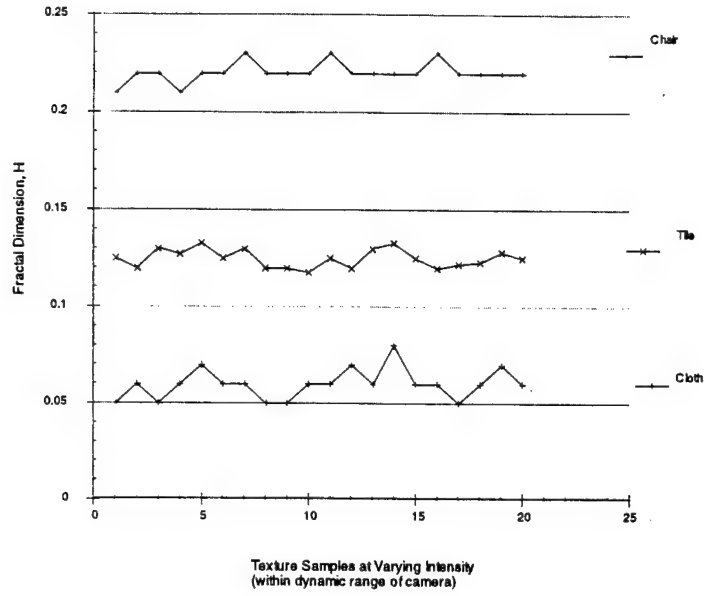


Figure 4: Variation of Fractal Dimension with Incident Light

5.4 Intensity Invariance

The texture images used in the classification experiments were available only in digitized form. As such, we chose 3 additional textures, ceiling tile, cloth and chair-cloth, to conduct the lighting sensitivity experiment. For each of the three textures we captured 10 different images of each texture while uniformly varying the incident light from a incandescent lamp within the linear range of the camera. The fractal dimension of each texture was computed for each intensity setting. The variation is shown in Figure 4. We observe that the estimated fractal dimension is a line almost parallel to the light axis indicating that the IFBM model is invariant to changes in incident light intensity within the linear range of the camera.

6 Conclusions

Texture classification is one of the first steps in image segmentation and is useful in locating potential sites where a target object might be located in an image. In this paper, we have presented a new fractal model for natural texture classification. We have developed a new fractal dimension estimator. We simplify the classification computations by using the ratio of the autocorrelation and the variance which is directly proportional to the fractal dimension.

We have compared the performance of this estimator to other techniques discussed in literature. We have found that the IFBM model performs best when the data fits the expected model. There are other fractal estimators which give better performance but these methods have higher computational complexity. We have shown that the developed model is invariant to changes in incident light intensity. We are investigating the

performance of this model when specular highlights are present in the images. We are also conducting experiments to verify the invariance to viewing direction. We are also attempting to develop a robust image segmentation algorithm.

References

[1] A. Rosenfeld and M. Thurston, "Edge and curve detection for visual scene analysis," *IEEE Transactions on Computer*, vol. 20, pp. 562–569, 1971.

[2] B. Julesz, "Visual pattern discrimination," *IRE Trans. Inform. Theory*, vol. 8, no. 2, pp. 84–92, 1962.

[3] R. Kashyap and K. Eom, "Texture boundary detection based on the long correlation model," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 11, pp. 58–67, 1989.

[4] R. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classifcation," *IEEE Transactions of Systems, Man and Cybernetics*, vol. 3, no. 6, pp. 610–621, 1973.

[5] K. Laws, "Classification based tracking of objects and materials," Tech. Rep. 443, SRI International, July 1988 1988.

[6] R. Kashyap and A. Khotanzad, "A model based method for rotation invariant texture classification," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 8, pp. 472–481, 1986.

[7] B. B. Mandelbrot and J. W. V. Ness, "Fractional brownian motions, fractional noises and applications," *SIAM Review*, vol. 10, pp. 422–437, October 1968.

[8] A. P. Pentland, "Fractal-based description of natural scenes," *IEEE Trans on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 661–674, November 1984.

[9] C.-C. Chen, J. S. Daponte, and M. D. Fox, "Fractal feature analysis and classification in medical imaging," *IEEE Trans on Medical Imaging*, vol. 8, pp. 133–142, June 1989.

[10] J. Keller, S. Chen, and R. Crownover, "Texture description and segmentation through fractal geometry," *Computer vision,graphics and image processing*, vol. 45, pp. 150–166, 1989.

[11] S. Peleg, J. Naor, R. Hartley, and D. Avinir, "Multiple resolution texture analysis and classification," *IEEE Trans on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 518–523, July 1984.

[12] H. G. E. Hentschel and I. Procaccia, "The infinite number of generalized dimensions of fractals and strange attractors," *Physica 8D*, pp. 435–444, 1983.

[13] T. Lundahl, W. J. Ohley, S. M. Kay, and R. Siffert, "Fractional brownian motion: A maximum likelihood estimator and its application to image texture," *IEEE Trans on Medical Imaging*, vol. 5, pp. 152–161, September 1986.

[14] K. Falconer, *Fractal geometry: mathematical foundations and applications*. England: John Wiley and Sons, 1 ed., 1990.

[15] H. Potlapalli, *Outdoor Landmark Recognition using Fractal Based Vision and Neural Networks*. PhD thesis, North Carolina State University, 1994.

[16] P. Brodatz, *Textures: A photographic album for artists and designers*. New York: Dover Publications, 1966.

[17] H. Loh, *Natural scene description using fractal features*. PhD thesis, North Carolina State University, 1991.

[18] A. Law and W. Kelton, *Simulation Modeling and Analysis*. New York: McGraw-Hill, 1982.

[19] R. Conners, M. Trivedi, and C. Harlow, "Segmentation of a high resolution urban scene using texture operators," *Computer Graphics, Vision and Image Processing*, vol. 25, pp. 273–310, 1984.

[20] R. C. Luo and H.-H. Loh, "Natural scene understanding using fractal features," in *IEEE Industrial Electronics Conference*, (Kobe, Japan), 1991.

A Hybrid Vision System for Meaningful Landmark Isolation and Recognition

Mir Azam and Ren C. Luo

Department of Electrical and Computer Engineering
North Carolina State University
Center for Robotics and Intelligent Machines
Box 7911, Raleigh, NC 27695
e-mail: msazam@eos.ncsu.edu

Abstract

Vision based navigation is the primary motivation for the research effort narrated in this paper. In order to achieve reactive and reflexive motion for navigation, a robot must be equipped with sensory system capable of performing rapid analysis of sensory data such that the controller is fed with the current information about the environment instantly. If this information is visual, as it is for human beings in many cases, then a good image processing technique becomes essential for the mobile platform. The information acquired need to be segmented into appropriate divisions and the segmented information need to be recognized properly. Our architecture for the vision system separates these two major tasks—segmenting image information and recognizing the segmented information. There are two algorithms presented—one for region transmission in the global image, and the other for neural network based learning mechanism.

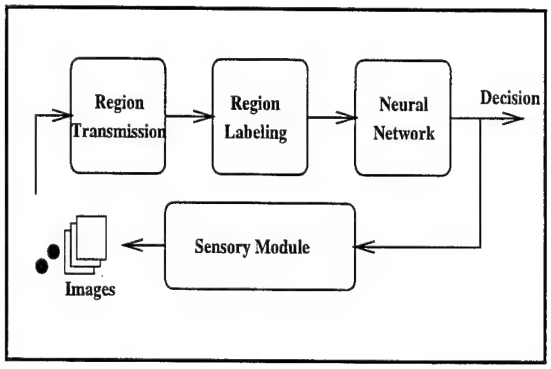


Figure 1: *Proposed system architecture.*

1 Introduction

The vision system described is intended to be an integrated system capable of not only isolating regions of interest, but also recognizing certain patterns or landmarks in the regions of interest. The reason the object locatlization and recognition are separated into defferent modules is due to the presence of the massive amounts of information in a typical input image [1, 2]. A typical image of natural scene would have many objects and landmarks present. Trying to analyze all of these possibilities is a wastage of the computation time because the landmarks of interest for this research are of very specific types. Namely, we are only concerned with street signs or uniform regions. Therefore, a segmentation algorithm capable of isolating uniform regions as a preprocessor to the neural network recognition system serves great potentials to succeed in recognizing landmarks of interest if they are present in the camera view.

Information reduction in images is essential to achieve any real time response of a mobile robot [3, 4]. Many segmentation methods are availbale in the literature including rule based segmentation [5], recursive splitting [6, 7], dynamic thresholding [8, 9] and others. In all the cases, segmentation is treated as a single phase problem where all the image data is manipulated at the same level and segmented regions are generated. In this paper, a multi-phase segmentation algorithm is introduced where image data is reduced without region labeling and later on analyzed to label them. In addition, a

neural network learning module is added to explain the segmented data.

Based on the fact that reducing information via isolating only possible landmark cadidates is beneficial, we propose the system architecture for the sensory system of a mobile robot similar to Figure 1. A camera would take pictures of the environemnt in the mobile platform's navigation path. These images would be processed through the segmentation algorithms in order to isolate the significant and homogeneous regions from the cluttered environment. The transmission of potential regions allows the image to be segmented into different clusters excluding the background noise. Some of the noise is, however, introduced by the region transmission algortithm. At the second level of the segmentation, a connected comopnent analysis labels the transmitted regions separately and thus filters out some potential noisy regions transmitted trough the previous phase. At the next level, a well trained neural network is fed by masking the image according to the required input by the neural network. And if a certain traned pattern is presented in the image, one of the masks fed to the network would recognize the pattern successfully.

2 Transmission of Regions

The purpose of this algorithm is to preserve data of interest in the image and filter out the rest of the information. We would like to perform this task successfully without the need for a heavy computational or methodical complexity. The intention initially was to complete the entire process upon one run on the image data.

The idea of image translation was incorporated with the principles of digital latches to obtain a fast resulting region growing process. A latch is a digital circuit gate or memory element which is capable of holding on to data upon clocking. The memory element mostly thought of is usually an edge triggered flip-flop which is triggered by either the leading or the falling edge of the clock driving the input to the gate. When looking at the timing diagram of a flip-flop, it is assumed the input data is stable enough to pass through the edge of the clock. Therefore, the data input at the edge of the clock is stored in the output bit of the flip-flop. Evidently, a change in the input data during the rest of the clocking period does not influence the stored bit.

A latch, on the other hand, is a memory element which allows the input data to pass through as long as the clock is high(for a leading edge triggered latch) or low(for a falling edge triggered latch) [10]. There is only a gate delay associated between the clocking and the data transmission period. Figure 2 shows an edge triggered D-latch and the timing diagram associated with the element.

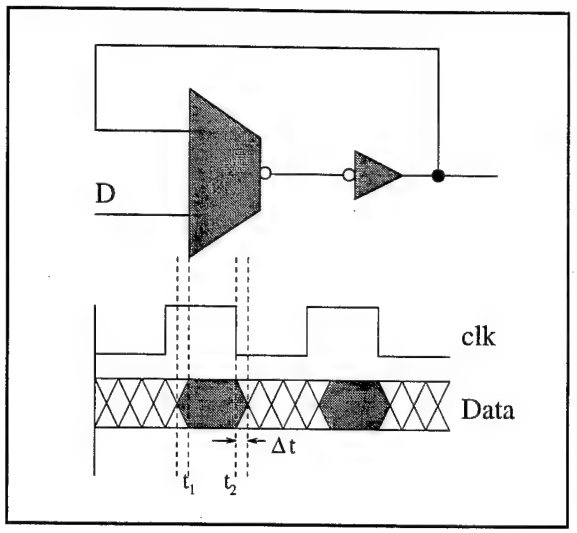


Figure 2: *A D-latch and the associated timing diagram showing data throughput only during clocking.*

A D-latch is made up of a 2:1 MUX driving an inverter at its output which is fed back as one of the inputs to the MUX. As it can be seen in Figure 2, the clocking occurs from times t_1 to t_2, and the data is passed through from time instances $t_1 + \Delta t$ to $t_2 + \Delta t$, therefore, keeping the data transmission interval the same, $t_2 - t_1$.

Analogous to the latching concept, the input data to the latch is considered as the image data, and the clocking initiation or termination is performed through the dissimilarity or similarity criteria of the algorithm. The similarity criteria is discussed separately, but for now it can be assumed that the desired threshold for adjacent pixels matching in the translated images are exceeded and a clock edge to the latch is initiated. If a high pulse is considered to be the clocking interval, then the first crossing of the threshold can be considered as the rising edge of the clock and the second fulfillment of the dissimilarity criteria can be considered as the falling edge of the clock.

For gray-scale images, where the signal strength at any point is determined by the gray level assignment of the pixel of the particular location, high dissimilarities are caused by high derivatives in the

data. High derivatives are caused by edges and the magnitudes of these high derivatives are determined by the edge strength. Therefore, the edges represent the rising and the falling edges of the clock to the latches.

The goal of the whole process is to preserve the data while the clock is high and filter out the rest when the clock is low. One of the first issues to consider is to determine how the clock pulses would be assigned according to the potential edges in the image. We define the filtering procedure by rows in the image plane. The data is passed through one row at a time, every pair of edges bounding potential edges. Upon conducting this data pass through the various latches in the image plane in every line, the significant regions in the image are automatically isolated from the background and other insignificant regions. In Figure 2, it shows the procedure of independently filtering data row by row in the image plane.

The edge strength or the clocking of the region growing process needs to be mathematically defined. The first derivative of the gray-scales are the the quantity defining edge strength between any two pixels in the image. If the two locations considered are (r_1, c_1) and (r_2, c_2), and their corresponding gray level assignments are g_2 and g_1, then the first derivative is defined to be:

$$E = \frac{g_2 - g_1}{\sqrt{(r_2 - r_1)^2 + (c_2 - c_1)^2}} \quad (1)$$

In order to find the clocking of the latches to be all positive, the above quantity is thought of as an exponential which provides a positive clocking at all possible edge locations as follows,

$$E_c = exp\left[-\left(\frac{g_2 - g_1}{\sqrt{(r_2 - r_1)^2 + (c_2 - c_1)^2}}\right)^2\right] \quad (2)$$

According to the latching concept, data is allowed to pass through only if there is a high enough clocking or edge strength at some location in the image. For our purposes, similar situation occurs. The exponential above describes all clocking in the positive direction, but represents the strength in the opposite way than it should be. Therefore, the actual data transmission coefficient is defined to be the reciprocal of the exponential as below,

$$TR = \frac{1}{exp\left[-\left(\frac{g_2 - g_1}{\sqrt{(r_2 - r_1)^2 + (c_2 - c_1)^2}}\right)^2\right]} \quad (3)$$

This is the value that is thresholded and used as the input to the clock of the latch that would let the image data pass through. If the thresholding function is expressed as $Th(x)$, then the image, $I(r, c)$ is filtered to the output $R(r, c)$ as follows:

$$R(r,c) = \bigcup_{allr,c} \frac{I(r,c)}{exp\left[-\left(\frac{g_2 - g_1}{\sqrt{(r_2 - r_1)^2 + (c_2 - c_1)^2}}\right)^2\right]} \quad (4)$$

Once the image is reconstructed from the original and the edge images, it is possible to recalculate the edge strengths from the output and the input images only. By expressing the exponential term in terms of the input and the output images, it can be shown that,

$$exp\left[-\left(\frac{g_2 - g_1}{\sqrt{(r_2 - r_1)^2 + (c_2 - c_1)^2}}\right)^2\right] = \frac{I(r,c)}{R(r,c)} \quad (5)$$

In other words, the edge strength can be written as the square root of the difference of the natural logs of I and R.

$$E = \sqrt{ln(R) - ln(I)} \quad (6)$$

This is a compact expression for proceeding with the reconstruction of the derivative image. This equation, therefore, can be an excellent measure of any iterations introduced in this algorithm. As one may expect, data can leak through the latches if they are turned on and off by false alarm such as noise in the input. But, knowing the rate of change of the edge data, one can easily learn when there is no significant change in the reconstructed edges, hence indicating no further data or region leakage in the global context.

3 Connected Components

The technique described in the previous section transmits the significant regions of the image, but also transmits noise instead of and added to the homogeneous regions. The reason for that happening is because the clocking of the latches are not controlled. In addition, the transmitted regions are transmitted row by row and there are no labels associated with the regions. In order to tackle these two problems, there is a second phase of the segmentation algorithm which determines the labels of the regions and also filters out small noisy regions transmitted through by the previous phase.

This phase is the connceted component based region growing in our system.

The region growing at any point in the image is based on the neighborhood type and the similarity criteria of the neighbors. For our purposes, we used an 8-connected definition of neighborhood. The central pixel is the pixel under investigation, and the other eight pixels of the 3×3 area are the neighbors of that particular pixel of interest at the center.

The process of region growing is similar to threading similar components with a needle and a string. All the pixels in the neighborhood belonging to the region where the central pixel belongs to are picked up by this conceptual needle and they are placed along the thread following the needle. The popular process of such exploration in the neighborhood is usually done in a stack manner where each one of the neighboring pixels in the same region are placed along a stack, and the next step for region growing is to move down one pixel or item along the stack and look at that particular pixel's neighbors [6]. This particular approach essentially accomplishes the same task, except that instead of going through the trouble of looking at all the members in the neighborhood, we jump to the neighbors of the first similar neighbor.

If the image is expressed by $I(r, c)$ and the center of the kernel is expressed by $k(x, y)$, then the center of the kernel for the threading process becomes the following expression:

$$k(x,y) = \bigcup_{x-1,y-1}^{x+1,y+1} [S\{I(m,n), k(m,n)\}] \qquad (7)$$

The $S\{I(m,n), k(m,n)\}$ is the similarity criteria that are discussed later in this paper. This kernel position is dynamic within the location of the current neighborhood. This is also a recursive process since the neighborhood location, and hence the kernel location, is updated with all of the satisfied similarity outcomes chronologically. This process is, furthermore, continued in a top-to-bottom and left-to-right raster scan method until the end of the data set. The complete run of this procedure, assuming the kernel location is calculated by equation 7, can be expressed as follows:

$$L(x,y) = \bigcup_{(0,0)}^{(I_{xdim}, I_{ydim})} NI(x,y). \qquad (8)$$

The $L(x, y)$ term in the above expression is the labeled image and the $NI(x, y)$ term is the neighborhood at a certain (x, y) coordinate in the image.

The strength of this approach is that at any point and time of the exploration of the image, only one similar pixel of the central pixel in the neighborhood is sought for, and when the similar pixel is found, the search begins all over again in the new neighborhood—keeping the newly found similar neighbor that initiated this new search as the central pixel for this search. The backward move to get to the "other" neighbors in the previous neighborhoods begins only when no other pixels in the current neighborhood can be found that can meet the similarity criteria.

3.1 The Thresholding Criterion

The similarity criteria is important for determining the shapes of the regions grown by this algorithm. It is an issue of debate as to what defines similar pixels, or pixels belonging to the same regions [5] [7]. Since there are no prior knowledge assumed to be known about the shapes of these regions, *any* definition of similarity could be just as good. But, there are some assumptions that can be taken into consideration such as the uniform source of light. It is natural that occlusions and shadows caused by objects present or absent in the image can cause a severe shade difference in different parts of the same region. But, from a viewer's point of view, these differences can indeed cause one to assume that subregions are present in the same region or object. The central concept of similarity is, however, based on the derivative in the neighborhood along the directions of the eight neighbors in the image.

The derivative mentioned here is a multidimensional derivative. Since there are eight neighbors chosen for each center pixel under study, there are eight possible directions where the derivative can be measured. The mathematical expression for the derivative in a gray-scale image is the following:

$$S\{I(x,y), k(x,y)\} = \Delta g = \left| \frac{g_n - g_m}{n - m} \right| \qquad (9)$$

Here, n and m are the two pixels' locations between which the gray-scale derivative is being calculated. g_n and g_m are their corresponding gray levels. In our calculations, the n^{th} pixel is always the central pixel and the m^{th} pixel is the m^{th} neighbor of the central pixel. Eight such derivatives are calculated as needed in a recursive manner. Since we are always considering immediate neighbors, and since all

the neighbors in all different directions have equal importance and equal probability of the direction of the growing region, the quantity $m - n$ is always 1. Therefore, The derivative or the similarity criterion can be even simpler:

$$\Delta g = |g_n - g_m| \quad (10)$$

The thresholding could still be a potential problem truly affecting the region growing even by setting up the similarity criterion, or the derivative strength, in an absolute manner. There is a need for normalization of this quantity. The best way to normalize this quantity is to normalize on the basis of the average intensity level of the image. If the average gray level of a pixel in the image is g_{avg}, then the similarity criterion is modified to be the following:

$$\Delta g = e^{\left(\frac{g_n - g_m}{g_{avg}}\right)^2} \quad (11)$$

This transforms the entire data set into a set bounded by $[0, 1]$. The connectivity-strength need not be a completely unknown quantity if we can normalize it through such a method. The uniformity in the data spread by applying the exponential are the connected regions, and the sharp valleys produced are their connecting boundaries. The thresholding is done if $\frac{g_n - g_m}{g_{avg}}^2 > 1.0$ and not thresholded otherwise. Upon thresholding the image, a concept called the similarity histogram is used to determine valid regions.

3.2 The Similarity Histogram

The similarity histogram is a measurement of the pixel distribution according to the label of each individual pixel. The conventional histogram represents a statistical model for the gray level distribution in the image. Here, for constructing the similarity histogram, gray levels are no longer important, rather the region growing based pixel labels are the data elements for the construction of this histogram.

Similar to the conventional gray-level histogram, the similarity histogram is a statistical measurement of the pixel distribution in the image. This distribution enables one to decide which regions, or threaded pixels, are the dominant regions in the image. Small strings of pixels, even though they may meet the similarity criteria, are most unlikely to represent any valid region in the image.

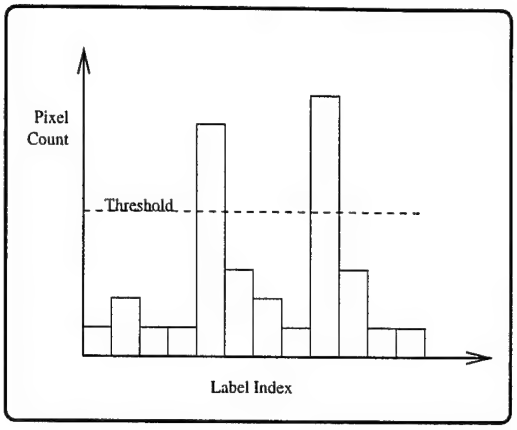


Figure 3: *The concept of Similarity Histogram.*

Thresholding the regions based on this similarity histogram is also another area to explore. In general, a region is significant if it covers a certain area in the picture. The most difficult decision to make is the one where it needs to be said that a particular region is indeed large enough to be a valid region. In general, according to the connected component theory, all pixels in the image are characterized with some label. But, if the labeled and assumed region is merely a discontinuity in some valid region, that discontinuity should not be considered as a separate region. It has been considered in this research effort that once a region consists of at least 10% of the total area in the image, that can be seen as a valid region. Mere discontinuities, this way, are meshed within these valid regions. This technique of making decisions with the similarity histogram is powerful especially for natural scenes where there may be trees and forests, clouds in the sky, or other natural discontinuities present. Figure 3 is an example of how the region verification is achieved using the similarity histogram.

The conventional gray scale histogram can be mathematically expressed by the following equations. If the i^{th} gray level is assigned to m_i number of pixels in an $N \times N$ image, the i^{th} histogram component has a value of

$$H(i) = \frac{m_i}{N \times N} \quad (12)$$

This gives a dynamic distribution of gray levels over the entire image. The similarity or region histogram is very similar to this concept, except that the pixel gray-scales are replaced by the pixel labels. If the i^{th} label has l_i number of pixels assigned to it, all of which are by all means connected pixels, then

the corresponding similarity histogram component would be

$$H_s = \frac{l_i}{N \times N} \quad (13)$$

Therefore, the similarity histogram can be expressed as

$$H_s = \sum_{i=0}^{\#of labels} \frac{l_i}{N \times N} \quad (14)$$

This, of course, is applied for an $N \times N$ image. And similar to the conventional histogram method, the sum of all the components add up to 1. That is

$$\sum_{i=0}^{\#of labels} H_s(i) = 1.0 \quad (15)$$

What the similarity histogram actually contain is a probability distribution function among the regions labeled in the connected component analysis. The height of each of the elements in the histogram is analogous to the likelihood of a valid region in the image. It is understandable that the data distribution in the input set can have the connectivity criteria satisfied for various locations in the image. Most of these connected-pixel sets do not represent true regions. Therefore, this probability distribution function based on the labeling of the connected components enables us to filter out the invalid and insignificant regions.

4 The Neural Network Module

The first phase of the problem, the isolation of candidate landmarks, is solved by the segmentation algorithms in two steps. The second phase of the problem is selecting among the potential candidates the true candidates which have meaningful information embeded in them. Potential landmarks can be street signs such as the STOP sign, the DO NOT ENTER sign and other meaningful features. We propose a neural network recognition algorithm to solve this phase of the landmark recgnition problem. The neural network was already under development for optical character recognition. However, typical landmarks can also be considered as distinct patterns of data and as such included in the landmark recognition problem. There are some test results on ten landmarks— such as the STOP sign, the DO NOT ENTER sign, etc. —that proves that the neural network designed was robust and integrable with our vision system.

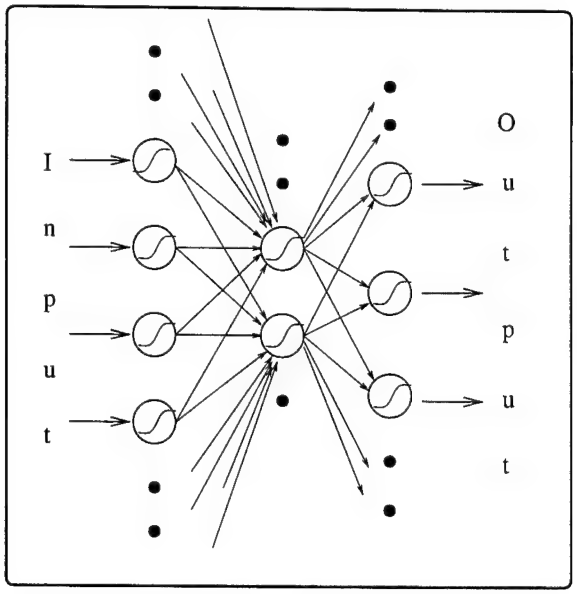


Figure 4: *Portion of the structure of the proposed fully connected network.*

4.1 Structure of Proposed Network

The structure that we propose in the optical character recognition problem as well as the is a multilayered feed-forward network with an input layer, an output layer and a hidden layer for weight adjustments. The input to our network are digitized images and therefore, the input is fed into individual nodes in parallel. That means that if we have an $n \times n$ image, the input layer of the network should have n^2 nodes.

A brief description of the structures of the individual layers is provided here.

4.1.1 The Input Layer

As discussed earlier, the input layer is considered to be set by the input data set or image. For the design purpose, the input layer was set to have 900 nodes capable of handling up to 30×30 pixels of image data. All of these nodes are connected to the hidden layer which is adjusted as follows.

4.1.2 The Hidden Layer

The hidden layer to our network consists of 5 nodes. Each one of the 5 nodes are connected with every input nodes(that is 900×5 neurons) and also with all the output nodes(that is 10×5 neurons)—giving a total of 4550 neurons in the network. The weights associated with these neurons are varied according

to the errors generated in each epoch of the training session. The final weight adjustments correspond to the minimum error suitable for all the input training sets.

4.1.3 The Output Layer

The output layer consists of 10 nodes each one of which represents a numeric digit in decimal mathematics. the output of these nodes refer to the similarity or dissimilarity of the testing pattern to the numbers 0 through 9.

4.2 The Mapping Function

The mapping function used to map the output of each node corresponding to the input to that particular node is chosen to be the sigmoid function. The equation describing the sigmoid function is as follows:

$$g(u) = \frac{1}{1+\frac{1}{e^u}} \tag{16}$$

There are reasons to choose such a mapping function. Note that the sigmoid function is a bounded function between 0.0 and 1.0. When the parameter u is small, the function gets closer to 0.0 and when the parameter is large, the value gets closer to 1.0. The function has a pretty steep derivative when the input to the function is close to 0. Therefore, this mapping function describes that as soon as we have a low input, the output gets close to 0, and the output is close to unity if we have a large input. this acts like a thresholding function but in addition to thresholding, it is sensitive to the weight of the positive or negative input to the function.

4.3 Symmetrical Weight Initialization

One of the issues to consider in distributing the network is a suitable position for the network to start. This is a critical point since if the initial position of the network is already close to giving the correct output for a given data set, the training would be brief. But how do we know about the weight distribution?

It is logical to assume that since the training and testing patterns to a network are completely unknown to the initial network, the inputs could be considered random. Any random statistical model would be, therefore, sufficient to initialize the network. We chose a zero mean Gaussian distribution with a variance of 0.5 to be the initialization function for the neuron weights. In general, if the input pixels are indexed by i, then the probability density function to distribute the initial weight would be,

$$p(w_i) = e^{\frac{-(w_i-\mu_i)^2}{2\sigma^2}} \tag{17}$$

4.4 Weight Adjustments

The weights of all the neurons are adjusted with respect to the errors that are generated by the input vectors in every epoch. The input node weights are dependent on the inputs to the system, while the hidden and the output nodes' weights are distributed and adjusted according to their inputs, the output of the nodes, and the error derivative of the node in question. A brief discussion on the issues are due here.

We use the chain rule of derivatives to calculate the derivative of the error with respect to the node weight,

$$\frac{\delta E}{\delta b} = \frac{\delta E}{\delta z}.\frac{\delta z}{\delta v}.\frac{\delta v}{\delta b} \tag{18}$$

Here $\frac{\delta E}{\delta z}$ is the derivative of the error with respect to the output of that node, $\frac{\delta z}{\delta v}$ denotes the derivative of the output of the node with respect to the weighted sum inputs, and the term $\frac{\delta v}{\delta b}$ expresses the derivative of the weighted sum input with respect to the output node.

Here is a brief explanation to each one of the terms involved. The error, for each one of the output nodes, is determined as the mean square error between the output and the target output of the node. The output is denoted by z in our discussion, and let us call the target t. Therefore, the mean square error is,

$$E = \frac{1}{2}(z-t)^2 \tag{19}$$

As we can see, the derivative term $\frac{\delta E}{\delta z}$ becomes,

$$\frac{\delta E}{\delta z} = z - t \tag{20}$$

Now we can focus on the second term where the output z is expressed as a function of the input v with the sigmoid function.

$$z = \frac{1}{1+\frac{1}{e^v}} \tag{21}$$

$$\frac{\delta z}{\delta v} = -\left(\frac{1}{1+\frac{1}{e^v}}\right)^2.(-e^{-v}) \tag{22}$$

This expression can be reduced to the following after algebraic manipulation:

$$\frac{\delta z}{\delta v} = \frac{e^v}{1 + 2e^v + e^{2v}} \quad (23)$$

As for the last term $\frac{\delta v}{\delta b}$ for a particular node, the relationship between the input to the node and the weight of the node is,

$$v = \sum_{i=1}^{N} b_i.y_i \quad (24)$$

And thus, this component of the chain rule has N sub-components.

The weight adjustments more or less follow this idea of successive derivatives. The specific discrete equations for weight adjustments are as follows:

If the outputs are denoted as V, the inputs as U, and the learning rate as η

$$w_{i+1} = w_i + \eta.w_i.U.w_i.V.(1 - w_i.V) \quad (25)$$

for the input and hidden layers, and for the output layer with a target of t

$$w_{i+1} = w_i + \eta.w_i.U.(1 - w_i.V).(w_i.V - t) \quad (26)$$

The symbol η denotes the learning rate of the network. Note that the error, the input and the output gain all are included in the weightings of the neurons.

5 Results

All the different aspects of the hybrid vision system have been tested and evaluated with natural scene images. At first, the region transmission algorithm was applied on several images with meaningful landmarks in them. The transmitted regions are shown in this section. Also, the connected component analysis yielded region labeling among the significant regions in the images. In addition, the neural network was trained on 10 different landmark patterns and recognized successfully. A brief description of the results are presented in this section.

5.1 Segmentation of Images

Figure 5 delineates the Transmission of Regions algorithm. As it can be seen, this algorithm was particularly successful in preserving natural landmarks

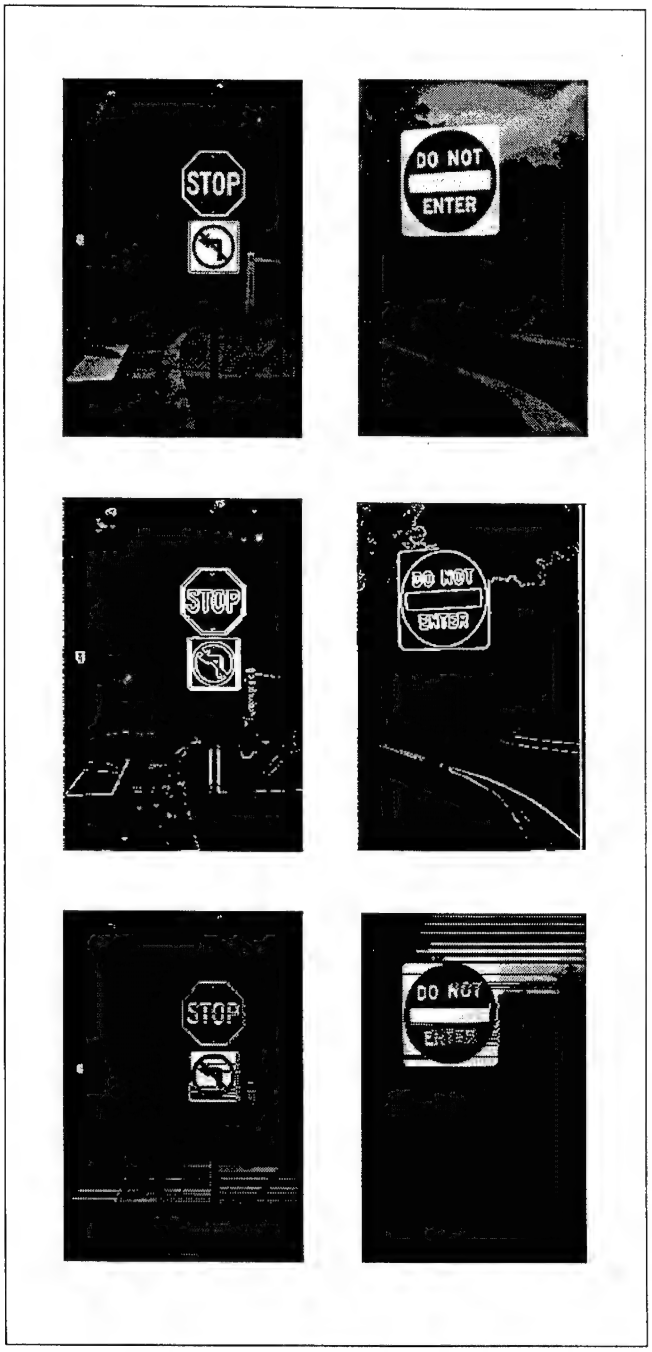


Figure 5: *Results of applying Region Transmission algorithm to natural scene images. A stop sign and a do-not-enter sign are used as examples here. The top row is the natural images, the middle row is their edge maps and the bottom row is their transmitted regions. The street signs have been isolated from the cluttered environment. Last but not the least, in the output image some transmission leakage is observed.*

in our everyday life. A lot of details in the landmarks are preserved whereas the background and other unnecessary information have been filtered out. But, due to the error in region-edge triggered clocking, the latches allowed some leakage from surrounding regions.

The other result is shown in Figure 6. This is the Connected component analysis based on the threading process. This process is particularly successful in distinguishing large areas present in the image. The major reason for that to happen is, in the similarity histogram criterion, it was decided that if the homogeneous region candidate is occupying at least 5–10% of the entire image, the validity of the region is preserved. Otherwise, the region is filtered out. Figure 6 shows region segmentation on a river bank separating the river the bank and the sky.

Both the algorithms are successful in segmenting regions in real images. The transmission of regions algorithm, compared to the connected component analysis, is computationally a lot less demanding. A good edge detector identifies the boundaries that need to be transmitted, and the second pass on the raster scan simply transmit the selected regions. The connected component analysis, however, speeds up from the traditional algorithm due to the threading process. It is important that candidate landmark regions from images be separated from the rest of the objects in the image, and these algorithms are effective tools for doing just that. According to the architecture of our vision system, the result of the segmentation prcedure is the input to the neural network.

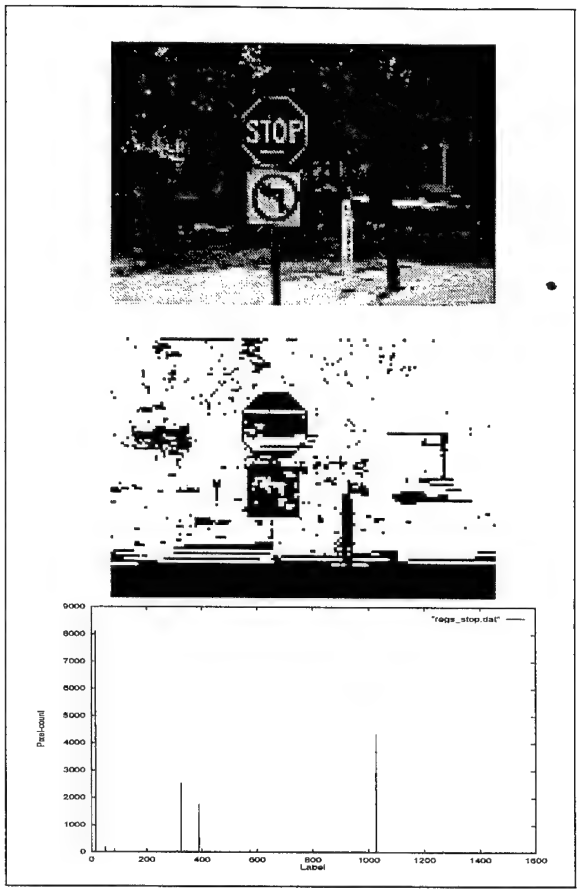


Figure 6: *Segmenting a natural scene image with connected component analysis (the corresponding similarity histogram is shown in the bottom image).*

5.2 Neural Network Output

The results for our network is listed here. But before listing the results, we need to mention the training method. There were 20 images that we had in our possession. We used 10 of these images— set of images with the street signs NO PARKING, DO NOT ENTER, LIGHT AHEAD, ONE WAY, PEDESTRIAN, RIGHT TURN, SCHOOL ZONE, STOP, ROAD CROSSING and YIELD was used to train the network— and the final set of the same patterns was used to test the network.

5.2.1 Individual Neuron Profiles

Each class of data created a particular network weight structure individually for recognizing each individual character. As we planned earlier, each character was supposed to have its own path in the weight space of the network. The way we described this path is by listing the final weight of each individual weights as a particular set of inputs are finished training. Let us call this as the neuron profile. The neuron profiles for all the training sets are expressed in Figure 7.

5.2.2 Comparative Neuron Profiles

It can be noted that the neuron profiles are different for each training patterns. Since there are a lot of weights to be plotted in the neuron profile plot, it is difficult to understand the difference in the neuron profiles. Therefore, a comparative plot between two neurons are plotted in Figure 8 to illustrate the uniqueness of the sensitivity pattern for each type of training set.

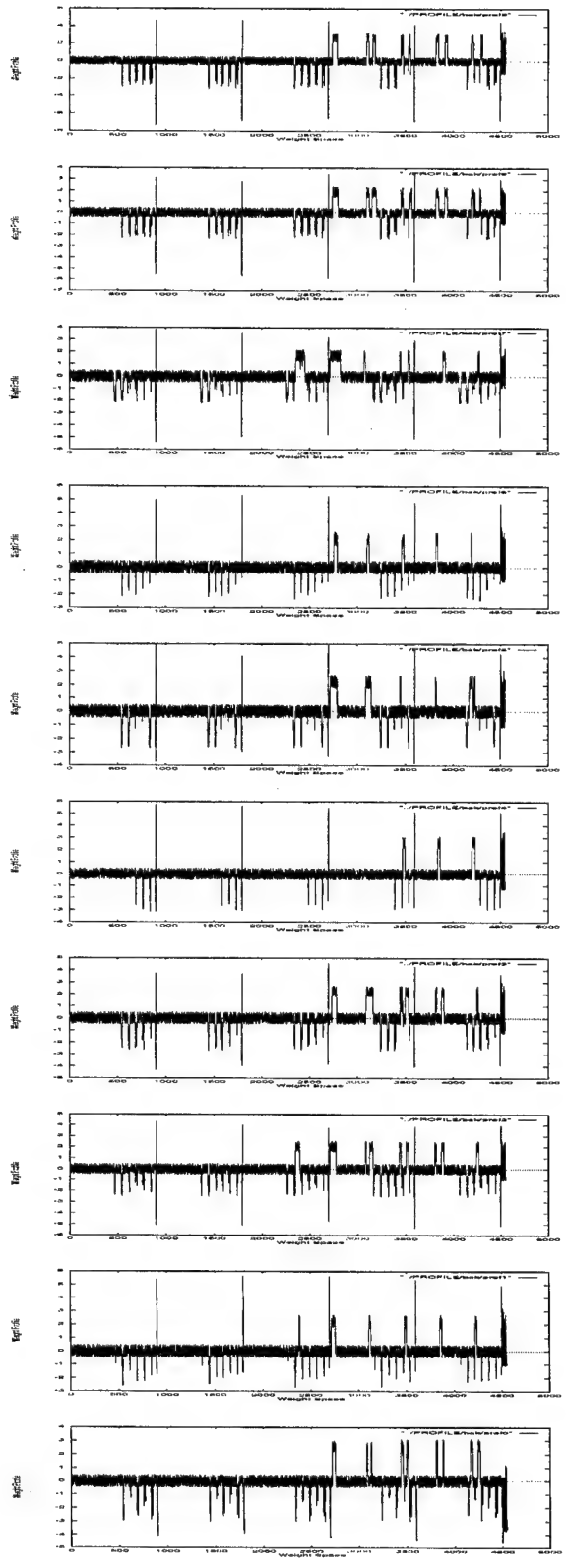


Figure 7: *Profiles of neurons(from bottom to top: no parking, do not enter, light ahead, one way, pedestrian crossing, right turn, school zone, stop, road crossing and yield). Horiz. axis: weight space, vert. axix: weight profile.*

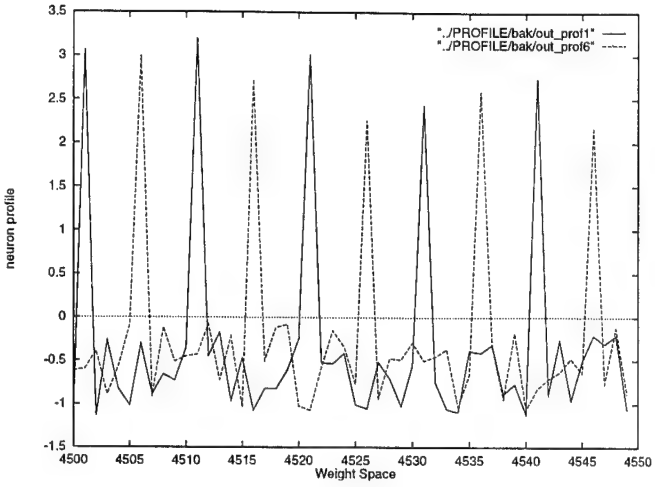


Figure 8: *Comparative profiles at the output layer of the network.*

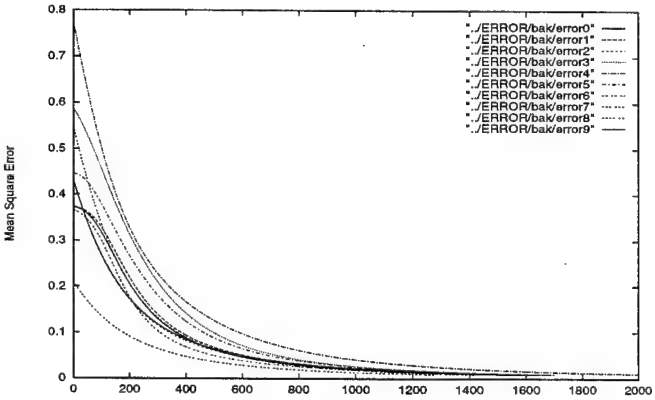


Figure 9: *Error Plots for all 10 output nodes. Horiz. axis: error, vert. axis; iteration.*

5.2.3 Network Output Error

The network Output Error is considered to be the mean square error between the output of the network and the desired output of the network. For example, we had decided that the sign "NO PARKING" should be responded by output 0, "DO NOT ENTER" should be described by output 1, etc. The other outputs in each case should be significantly lower than that. Each time we tested the network, and also while training it, we calculated the network mean square error and plotted them.

The equation to find the error is as follows:

$$MSE = \sum_{i=0}^{9} \frac{1}{2}(Output_i - Target_i)^2 \quad (27)$$

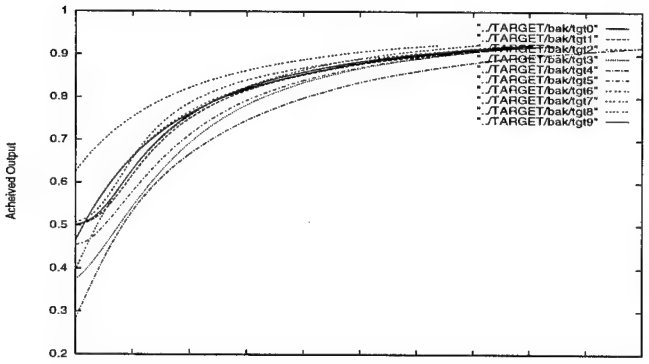


Figure 10: *Output Responses. Horiz. axis: output, vert. axis; iteration.*

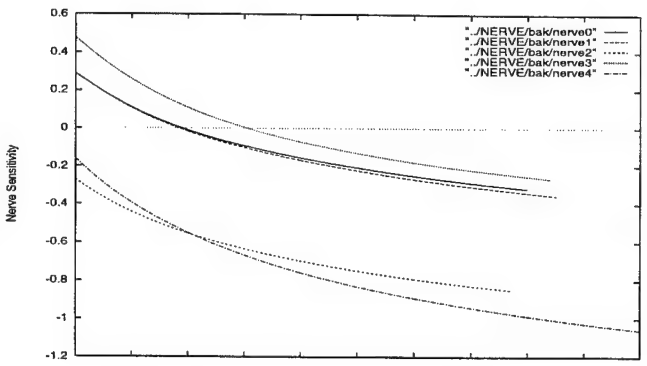


Figure 11: *Neuron paths different for each input vector with respect to training iterations.*

5.2.4 Generalization

A big problem in the network that we faced was the generalization of the network. We tried to train the network such that different images with the same sign would be recognized by the network without classification error. For this, we trained the network until the error at each node was less than 0.01%. As we tested on the test set, and also by feeding some of the training samples again, the network response between RIGHT TURN and SCHOOL ZONE are sometimes close. This is probably due to not having enough resolution in the images. But, on the average, the network recognized all characters of different fonts without miss-classification. Figure 11 is a good illustration of generalization being achieved for five of the signs tested in the laboratory. According to the figure, each one of the shown five pattern tends to follow a generalized path of its own.

6 Conclusion

The hybrid vision system proposed in the paper consists of three algorithms. The first two compose the region isolation and the third is the neural network learning. All three algorithms are developed separately but are designed to be integrated in the same system. Real image analysis is the strength of these algorithms. The methods described prove to be robust, efficient and accurate. The region transmission and labeling process meete the goal of isolating significant regions adequately as illustrated in the paper, and the neural network learning is accurate enough to distinguish among 10 different aptterns. Therefore, this approach towards developing a vision based navigation system is promising.

7 Future Work

Even though the techniques narrated are individually successful, their integration is a sensitive issue at this point and time. There have been numerous experiments showing successful transmission of region, successful region labeling and successful neural network performance for recognition. The outputs of transmitted regions have been separately fed into the input of the labeling module, the segmented images are recognized with the neural networ, but the complete loop of the system architecture is not complete yet. We envision to tie these loose ends in the near future and transport the software to a mobile platform in the laboratory for autonomous navigation.

References

[1] C. Tsikos and R. Bajcsy, "Segmentation via manipulation," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 306–319, 1991.

[2] M. Blancard, "Road sign recognition: a study of vision based decision making for road environment recognition," *Vision Based Vehicle Guidance*, 1992.

[3] D. Pomerleau, *Neural Network Based Autonomous Navigation.* Kluwer Academic Publishers, 1990.

[4] C. Chen, M. Trivedi, M. Azam, and N. Lassiter, "Simulation, animation, visualization and interactive control of a tracked mobile manipulator," *SPIE International Conference, Sensor Fusion VII, Boston, MA*, 1993.

[5] M. Levine and A. Nazif, "Rule based image segmentation: a dynamic control strategy approach," *Computer Vsion, Graphics and Image Processing*, vol. 32, pp. 104–126, 1985.

[6] R. Ohlander and K. P. et. al., "Picture segmentation by recursive region splitting," *Computer Vision, Graphics and Image Processing*, vol. 8, pp. 313–333, 1978.

[7] H. Raafat and A. Wong, "A texture information directed region growing algorithm for image segmentation and region classification," *Computer Vision, Graphics and Image Processing*, vol. 43, pp. 1–21, 1988.

[8] M. Spann and C. Horne, "Image segmentation using a dynamic thresholding pyramid," *Pattern Recognition*, vol. 22, no. 6, pp. 719–732, 1989.

[9] A. Perez and R. Gonzalez, "An iterative thresholding algorithm for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 6, pp. 755–760, 1987.

[10] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design.* Addison-Wesley, 1994.

[11] H. Potlapalli and R. L. et. al., "Translation and scale invariant landmark recognition using receptive field neural networks.," *International Conference on Intelligent Robots and Systems.*, 1992.

[12] M. Azam and et al., "Outdoor landmark recognition using segmentation, fractals and neural network," *Advanced Research Project Agency Image Understanding Workshop*, 1996.

[13] M. Spann and R. Wilson, "A quadtree approach to image segmentation which combines statistical and spatial information,," *Pattern Recognition*, vol. 18, no. 3, pp. 257–269, 1985.

[14] A. R. Rodriguez and O. Mitchell, "Image segmentation by successive background extraction," *Pattern Recognition*, vol. 24, no. 5, pp. 409–420.

Multilayered Fuzzy Behavior Fusion for Real-Time Reactive Control of Systems with Multiple Sensors

Steven G. Goodridge, Michael G. Kay, *Member, IEEE,* and Ren C. Luo, *Fellow, IEEE*

Abstract—**Fuzzy linguistic rules provide an intuitive and powerful means for defining control behavior. Most applications that use fuzzy control feature a single layer of fuzzy inference, mapping a function from one or two inputs to equally few outputs. Highly complex systems, with large numbers of inputs, may also benefit from the use of qualitative linguistic rules if the control task is properly partitioned. This paper presents a modular fuzzy control architecture and inference engine that can be used to control complex systems. The control function is broken down into multiple local agents, each of which samples a subset of a large sensor input space. Additional fuzzy agents are employed to fuse the recommendations of the local agents. Real-time implementation without special hardware is possible by using singleton output values during fuzzy rule evaluation. A development tool is used to translate a fuzzy programming language off-line for fast execution at run time. Using this system, a multilayered fuzzy behavior fusion based reactive control system has been implemented on an autonomous mobile robot, MARGE, with great success. MARGE won first place in Event III of the 1993 Robot Competition sponsored by the American Association for Artificial Intelligence.**

I. Introduction

THE success of fuzzy control is owed in large part to the technology's ability to convert qualitative linguistic descriptions into nonlinear mathematical functions. Its ability to bridge the gap between human expert knowledge and the world of digital systems has led to its use in many consumer products. Most of these fuzzy control implementations, however, feature a single layer of inferencing between only a handful of inputs and outputs, leading one to suspect that fuzzy control might only be useful in simple systems such as these. This paper presents a multilayered fuzzy control architecture and inference engine that can be used to control a complex system: the reactive guidance control of a mobile robot. Given the complexity of a mobile robot, linguistic rules provide an essential tool for implementing the robot's control, without the need for a mathematical model. A complex control mapping is described that uses expert rules in a layered, modular architecture that combines a friendly, modular fuzzy programming language with a real-time fuzzy inference kernel. Fuzzy uncertainty is not propagated from one level to the next in the architecture, as this is not an important requirement for many types of reactive control applications.

Autonomous mobile robots must react to dynamic events in unstructured environments. Vision, odometry, ultrasonic and/or infrared rangefinders, and tactile sensors may be used together to sense the robot's world. The challenge of integrating this information in real time has led to a variety of reactive "behavior-based" control schemes for mobile robots. Much of this work was inspired by the layered control system and subsumption architecture developed by Brooks [1], [2] in which robots utilize multiple task-achieving behaviors which closely couple sensors to actuators. Connell [3] used distributed subsumption-based behaviors to enable a mobile robot to search through unmapped rooms to collect empty soda cans. In a recent competition between autonomous robots [4], the winning entries made extensive use of low-level reactive perception and control.

Real-time reactive control involves mapping sensor inputs to control signals quickly, usually involving little or no intermediate representation. Most feedback control systems, such as servomotor controllers, involve only one or two inputs. Linear gain parameters are often sufficient for determining the actuator output for such simple applications. But for mobile robots and other complex reactive systems, the size of the input space demands a more sophisticated control algorithm. This task can be made more manageable by breaking down the input space into a form suitable for analysis by multiple agents, each of which responds to specific types of situations, and then integrating or fusing the recommendations of these agents to produce the current output.

Agents can be designed to exhibit independent behaviors such as goal seeking, obstacle avoidance, and wall following. This is an important level of abstraction for system integration, and can serve as a guide for the division of computation among distributed processors. Multiple behaviors also provide robust behavior and degrade gracefully. The combined effect of multiple behaviors is referred to as an emergent behavior, and, ideally, it will share the characteristics of the individual behaviors. Unfortunately, the development and integration of agent recommendations usually involves ad hoc algorithms that are application specific. Brooks' subsumption architecture uses finite state machines and hierarchical arbitration to select between multiple behaviors. This allows fail-safe

Manuscript received February 9, 1995; revised November 27, 1995. This work was supported in part by the National Science Foundation under Grant DMI-9322834.

G. Goodridge and R. C. Luo are with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911 USA.

M. G. Kay is with the Department of Industrial Engineering, North Carolina State University, Raleigh, NC 27695-7911 USA.

Publisher Item Identifier S 0278-0046(96)02372-6.

behaviors like obstacle avoidance to subsume goal seeking when necessary. Arkin [5] fuses independent behaviors called "schemas" with artificial potential-field techniques. In this paper, a modular method for developing reactive control behaviors using fuzzy control rules is described. Fuzzy control allows smooth generalization between multiple modes of operation. The function is a continuous surface, as opposed to many behavior-switching schemes, and follows the philosophy that similar situations should result in similar outputs. The system allows the integration of many more sensor inputs than do previous real-time fuzzy systems, without special hardware.

II. Fuzzy Control

Fuzzy control is based on fuzzy set theory, originally developed by Zadeh [6], [7]. The principle behind fuzzy set theory is that ambiguous information should be classified into sets that do not have crisp boundaries; hence the name "fuzzy sets." Given a measurement x, a fuzzy set A is said to contain x with a degree of membership defined as $\mu_A(x)$, where $\mu_A(x)$ can be any value in the continuous domain $[0, 1]$. Fuzzy sets are usually named after adjectives, such as *TALL*; the membership function described as $\mu_{TALL}(x)$ would therefore reflect the similarity between values of x and the contextual meaning of *TALL*. If x represented a person's height in centimeters, and *TALL* were used to classify "tall men," then *TALL* might have a membership function value equal to zero for heights below 150 cm, a value equal to one for heights over 185 cm, and a smooth curve of fractional values for heights between these limits. The degree of truth of a statement like "If Bob's height is $TALL\ldots$" is evaluated by calculating the value of the membership function for Bob's height. A fuzzy set may also be thought of as a distance metric for the comparison of quantitative data. Fuzzy set membership functions often take the shape of gaussian, trapezoidal, or sigmoidal functions.

Logical manipulation of fuzzy memberships requires the extension of crisp logic operations to operations on fuzzy sets. The three fundamental logical operations, intersection, union, and complement, have fuzzy counterparts popularly defined as follows:

(Intersection) (x is A) AND (y is B):
$$A \cap B = \min\{\mu_A(x), \mu_B(y)\}$$

(Union) (x is A) OR (y is B):
$$A \cup B = \max\{\mu_A(x), \mu_B(y)\}$$

(Complement) (x is NOT A):
$$\tilde{A} = 1 - \mu_A(x).$$

A fuzzy control function may be defined by using fuzzy sets as adjectives in a qualitative rule base. The effect of each rule inference is then proportional to the degree of truth of the fuzzy sets associated with it. When programming a fuzzy system with fuzzy rules, the system designer need only represent his or her own qualitative understanding of the problem.

A. Fuzzy Rule Evaluation

A fuzzy rule performs an inference with a certainty, or weight, dependent on the set operations in its antecedent. A fuzzy rule for a servocontroller may take a form such as: "If *error* is *SMALL* and Δ*error* is *ZERO*, then *output* is *SMALL*." This rule assigns the value of *SMALL* to the variable *output* with a weight determined by the intersection (i.e., the minimum value of two membership function evaluations) of the sets describing *error* and Δ*error*. The application of multiple fuzzy rules results in multiple output recommendations. In many fuzzy expert systems, antecedent weights are intersected with the output sets to describe the control output as a fuzzy variable. The output value has its own membership function, which is useful if it is to be used by other rules in a chain of fuzzy inferences. For control applications, however, defuzzification is necessary to obtain a singleton value that can be passed on to actuators. This is often achieved by taking the centroid of the output membership function. Unfortunately, the manipulation of fuzzy sets and the calculation of the membership centroid is computationally expensive when compared to ordinary linear control algorithms. Real-time fuzzy control systems that define outputs as fuzzy sets in this manner often rely on special hardware to perform rule evaluations [8], [9].

For most control applications, outputs do not need to take the form of fuzzy numbers. A much faster method can be used to evaluate rules by calculating the centroid of a set of singleton recommendations. If each rule i prescribes an output value of o_i with an antecedent certainty of w_i, then the output of a controller with N rules is calculated as

$$control\_output = \frac{\sum_{i=1}^{N} w_i \cdot o_i}{\sum_{i=1}^{N} w_i}.$$

To compare these two techniques, fuzzy output versus singleton output values, consider a very simple mapping with one input and one output. Fig. 1(a) defines the set to be used in this example. The first function in this example uses two rules with the following fuzzy output sets

(Function A) 1) If *input* is *ZERO* then *output* is *ZERO*.
2) If *input* is *SMALL* then *output* is *SMALL*.

Suppose we wish to calculate the output value of Function A for an input of value of 70. The first rule will have a weight of 0.3 and the second will have a weight of 0.7. Fig. 1(b) shows the result of the weighting operations on the respective fuzzy output sets. The output of Function A is the centroid of the two shaded regions, which is calculated to be 64.

Now consider the same function with output recommendations defined as the following singleton values

(Function B) 1) If *input* is *ZERO* then *output* is 0.0.
2) If *input* is *SMALL* then *output* is 100.0.

For Function B, the crisp centroid is much easier to calculate, as illustrated in Fig. 1(c). For an input value of 70, its output is $(0.3 \times 0.0 + 0.7 \times 100.0)/(0.3 + 0.7) = 70$. The singleton centroid scheme of Function B allows fuzzy rules to be used to perform real-time control functions on ordinary processor hardware. This method of rule evaluation is used for the reactive control system described below.

Fig. 2 shows the block diagram of a fuzzy rule. A rule performs a sum-of-products (i.e., intersection before union)

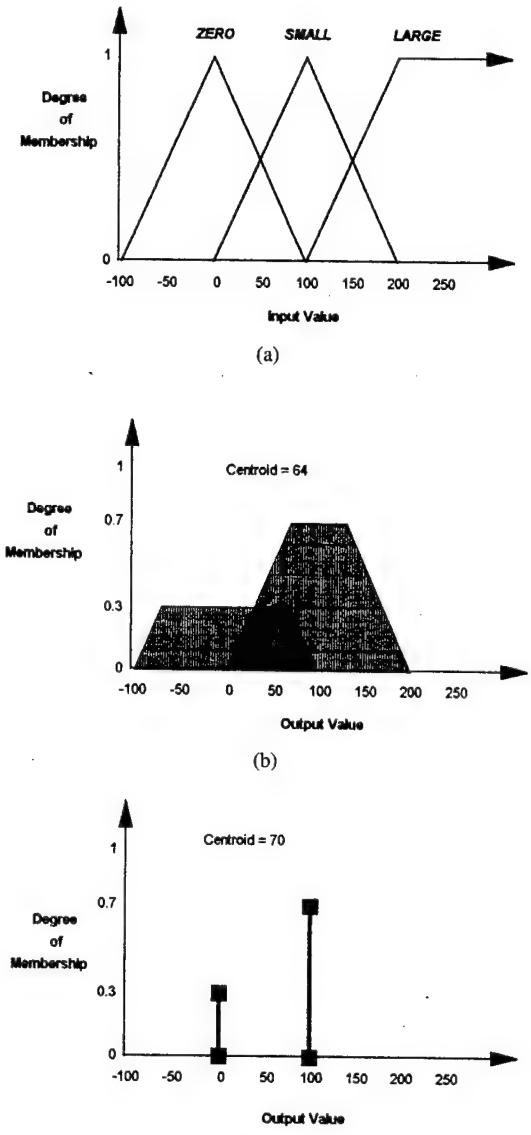


Fig. 1. Fuzzy rule evaluation example. (a) Membership functions. (b) Centroid for fuzzy output. (c) Centroid for singleton output.

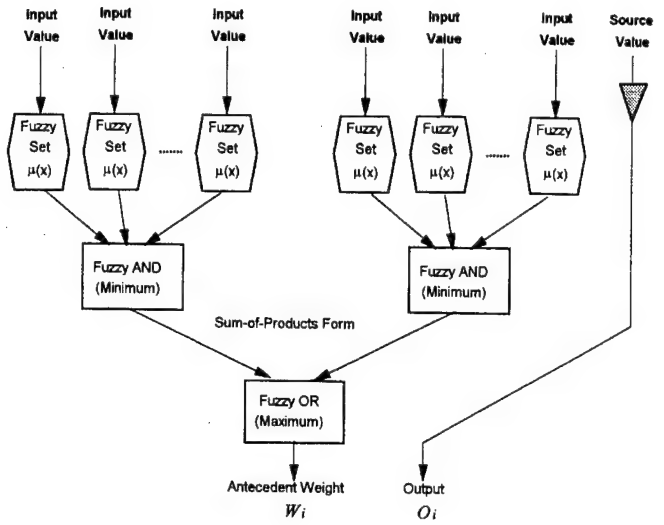


Fig. 2. Components of a fuzzy rule.

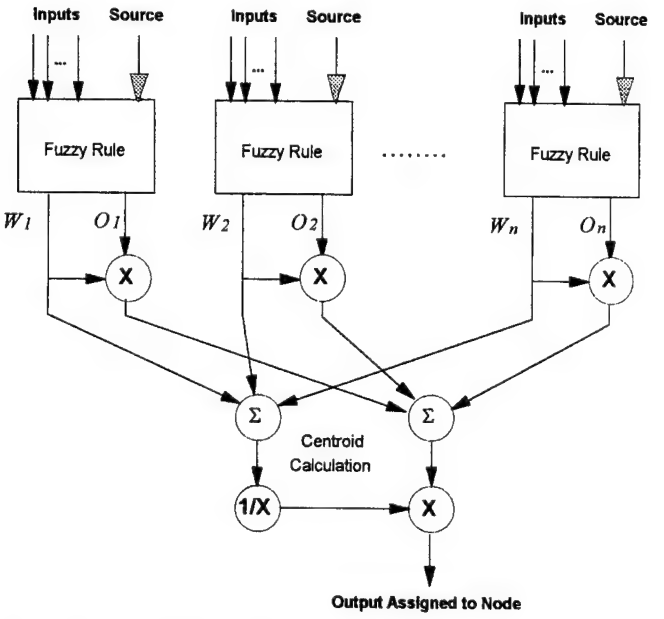


Fig. 3. Integration of fuzzy rules.

operation on the fuzzy set comparisons in its antecedent. The resultant weight is then associated with the source value for subsequent centroid calculations. Note that the source may be a fixed value, as is the case with most fuzzy control systems, or it may be a value passed from another operation, allowing a set of rules to act as a "fuzzy multiplexer" by blending recommendations from other sites according to qualitative terms. Fig. 3 shows the configuration of multiple rules for the evaluation of a fuzzy node. Note that black arrowheads denote set inputs, while gray arrowheads denote sources for output recommendations.

B. Effects of a Large Input Space

Most fuzzy control applications involve only a small number of inputs. This allows them to perform the entire inferencing mapping in one step. All of the fuzzy rules typically look at the same inputs and affect the same output. Suppose a controller for a system with N inputs is being designed, with each input i described by M_i fuzzy sets. A different rule may be written for every intersection of set descriptions that describes the N inputs. This exhaustive method yields a rule set of size $\prod_{i=1}^{N} M_i$.

Unfortunately, the number of fuzzy set evaluations in a rule base increases exponentially as more inputs are added to the controller. This results in an impractical computational load for systems like mobile robots with typically have high input dimensionality. It also makes it difficult for the programmer to manually define rules that span the entire input space. In order to keep the rule base manageable, some mobile robot control implementations have reduced the input space by throwing away what might otherwise be useful sensor data [8], or by first matching the data to a symbolic world model and extracting state variables [10], [11].

Rather than reducing a large input space by nonfuzzy means, the system described in this paper uses multiple fuzzy agents

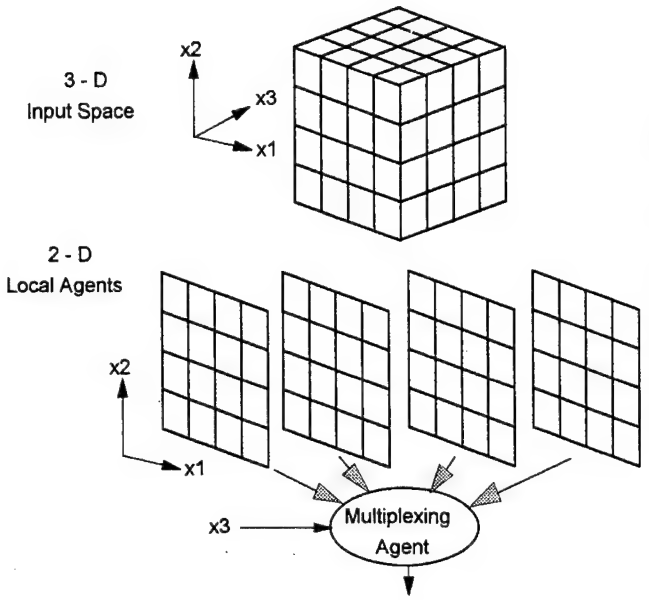


Fig. 4. Fusion of local agents by a fuzzy multiplexer.

to provide a computationally efficient means of processing the entire input space without an initial data reduction. Consider a symmetric N-dimensional input space where each input dimension is to be spanned by the same number, M, of fuzzy sets. If the fuzzy inference mapping is done in one step, the number of set evaluations that must be performed is NM^N. If the input space dimensionality is broken down for processing by local agents, fewer set evaluations must be performed. Suppose we employ M^n local agents, where each will be assigned the same $N-n$ inputs. These agents then may be fused by a fuzzy multiplexer that uses the n remaining inputs. The multiplexer will perform nM^n set evaluations, while each agent performs $(N-n)M^{(N-n)}$ evaluations. The total number set evaluations for this scheme is then $NM^N - n(M^N - M^n)$, which is less than NM^N. Thus, a large input space can be broken down in several steps for a considerable savings in computation. Fig. 4 shows a three-dimensional (3-D) input space processed by local agents and fused by a multiplexer. Note that there is no restriction to using only agents of uniform size or architecture; rather, the flexibility of using specialized agents of different sizes and/or types is an important part of this scheme. Each of the local agents, for instance, may be composed of additional agents in an overall hierarchical network of agents.

Fuzzy agents can also preprocess data for other nodes to use as inputs. This allows the input space to be transformed into another form that may be more useful for agents to react to. In most control applications, raw sensor data is not directly mapped to actuators without first transforming it for purposes such as filtering or to calculate error signals and derivative terms. This reduces the computational requirements for some applications and can make the system easier to specify manually by providing a higher level of abstraction than provided by the direct input data. Fuzzy rules can be easily configured to accomplish such preprocessing functions.

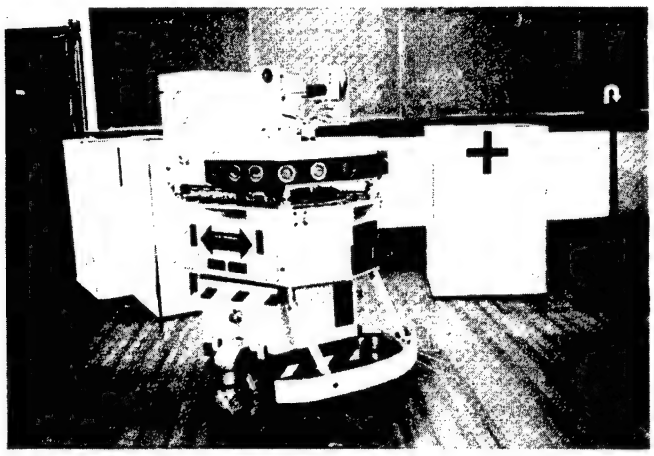


Fig. 5. Mobile robot MARGE.

III. Fuzzy Behavior Fusion

A user-friendly system has been developed for the design of multilayered fuzzy behavior-based control systems. This system can perform real-time control calculations for many applications using an ordinary personal computer. Fuzzy sets, nodes, defined constants, and fuzzy rules are defined by the programmer in a text file. This file is then translated off-line by a fuzzy translator into a data structure that enables fast execution at run-time. A Motorola 68 040 CPU running with a 20 MHz clock can typically process over 23 000 fuzzy rules per second using this data structure.

A. Implementation on the Mobile Robot MARGE

Multilayered fuzzy behavior fusion has been used to control MARGE (which stands for mobile autonomous robot for guidance experiments), an indoor mobile robot developed for experiments in autonomous guidance. A liberal assortment of sensors allows MARGE to measure information about itself and its environment. Fuzzy control is used to convert this information into reactive behavior. Fig. 5 shows MARGE, and Fig. 6 shows the installation of sensors on the vehicle. Vision, its longest-range sense, is used for selflocalization and for identifying objects of interest. For obstacle avoidance, the robot uses wide-angle and narrow-beam ultrasonic rangefinders. Tactile whiskers are used for feedback during manipulation tasks, and as a fail-safe for obstacle avoidance should objects be missed by the sonar. Encoders in the drive and steer motors provide dead reckoning and velocity feedback.

B. Fuzzy Behaviors

Examples of the behaviors used to control MARGE include goal seeking, obstacle avoidance, barrier following, and object docking. The obstacle avoidance behavior filters sonar data and suggests an appropriate steering and drive velocity given the presence of obstacles sensed by the vehicle's sonar. The goal seeking behavior generates the proper control values to attain a goal location, and the barrier following behavior stabilizes the vehicle's motion with respect to straight walls. The object docking behavior allows the robot to manipulate objects, which is usually difficult for autonomous systems in

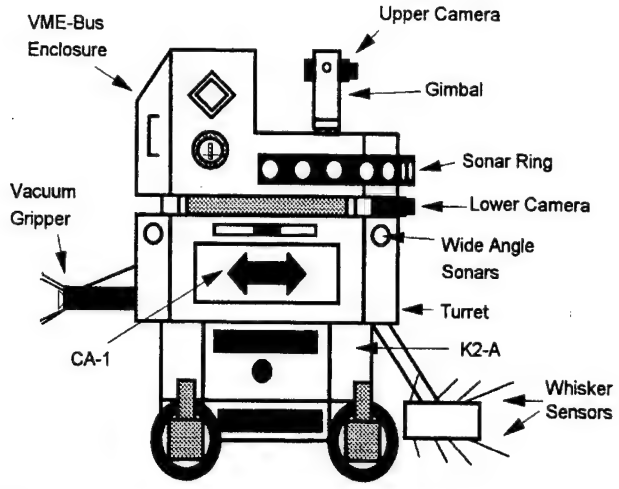


Fig. 6. Configuration of system hardware on MARGE.

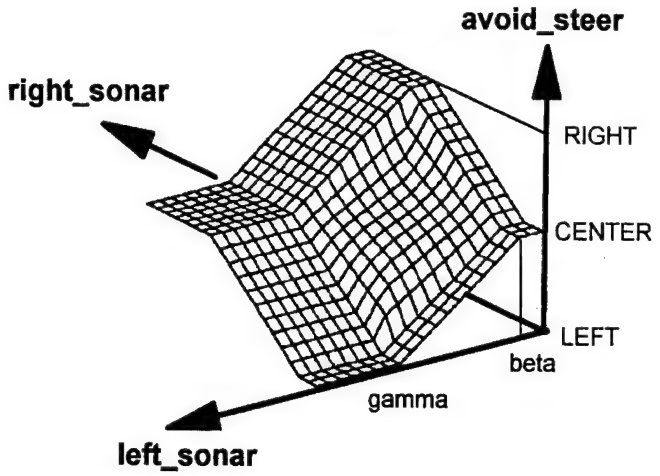


Fig. 7. Control surface for simple obstacle avoidance behavior.

unmapped environments. Additional behaviors can be added to adapt the vehicle's actions to new tasks and situations by adjusting the parameters of the other behaviors.

To illustrate the design of a behavior using fuzzy rules, consider a simple obstacle avoidance agent which tries to steer away from close obstacles sensed by sonar on either side of the robot. First, define the name of the behavior to be "avoid_steer" and define a fuzzy set to represent the adjective "close":

```
node avoid_steer
set close z beta gamma
```

The terms *beta* and *gamma* are values that correspond to the dynamic range of distances that the robot will avoid, with a strength of avoidance that decreases with range. The following four fuzzy rules are used to create this behavior:

```
if left_sonar is close
and right_sonar is not close
then avoid_steer is RIGHT
if left_sonar is not close
and right_sonar is close
then avoid_steer is LEFT
if left_sonar is close
and right_sonar is close
then avoid_steer is CENTER
if left_sonar is not close
and right_sonar is not close
then avoid_steer is CENTER
```

CENTER is a constant value equal to zero, while LEFT and RIGHT are typical steering velocities. These rules cause the robot to steer away from obstacles to one side or the other, depending on how close they are. If both ranges are equal, the robot proceeds straight ahead. Fig. 7 shows the smooth control surface that results from these rules.

While this example illustrates the smooth transition possible between fuzzy rules, it makes use of only two sensors. A much more competent obstacle avoidance behavior results from incorporating other sonars into the decision on how to turn. For example, the robot may not need to turn away from obstacles on its side unless its path was blocked. If a front sonar measurement is added as another input, the side avoidance distance can be increased as the front range decreases. This results in a "corner escaping" behavior. Diagonal sonars can be incorporated into behaviors in a similar manner, along with braking behaviors to slow and stop the robot when necessary.

Sometimes, two or more rules cancel each other out and the robot then faces symmetric indecision. This can occur when an obstacle is located straight ahead, for example, and no other obstacles exist on either side of the robot. MARGE breaks this kind of tie caused by symmetric indecision with a "frustration" agent, which increases in magnitude over time whenever the robot becomes blocked. An extra rule in the obstacle avoidance behavior adds a random steer value to its output when frustration becomes large. Thus, the robot senses when its basic rules fail to keep it moving and responds to the problem by using its motivational state.

The repulsion of the robot from obstacles may seem like a variation on the artificial potential fields method of obstacle avoidance. The most important distinction that can be made is that fuzzy behaviors are egocentric to the robot, and depend on its pose and motivational state, while potential fields often require an intermediate representation of the environment, such as a map. It is easy to create fuzzy behaviors that follow a wall and align to features that the robot's sensors detect. This allows the robot to escape from local minima by following a boundary, without requiring a high-level planner or world model. In order to provide the same performance using potential fields, one would have to program nonlinear, time-varying equations into the control program. The fuzzy rule approach allows bottom-up development without mathematical models. This makes it easier to implement sophisticated behaviors that directly reflect the programmer's expert knowledge.

C. Fusion of Behaviors

Suppose that, in addition to an obstacle avoidance behavior, a goal seeking behavior has been designed that steers the robot toward a goal location. Both behaviors must eventually be

fused into a single output to control the vehicle's motion. Many different schemes have been used for this type of fusion, including hierarchical switching [1]–[3] and weighted averaging [5]. Fuzzy behavior fusion combines hierarchical switching with weighted averaging by using fuzzy rules to perform the fusion operation. The fusion rules are flexible: sensor data, motivational state, and/or the values of the behavior outputs themselves can be used to determine the appropriate weight for each behavior. The following are four possible ways to blend or arbitrate between the recommendations from multiple sources.

1) Summation: A control signal recommendation for a goal seeking behavior might be simply added to the obstacle avoidance behavior. This method is the same type of superposition used in potential field techniques. In a PID controller, the proportional, integral, and derivative terms are summed. One must be cautious, however, as to what the worst-case combinations of values can be. If two behaviors each recommend a maximum value, their combined effect may be dangerous. Also, the recommendation of each behavior must be compatible with the recommendations of the other behaviors throughout the entire input space.

2) Weighted Averaging: To maintain a limit on the output magnitude, a compromise between agent recommendations can be reached using a weighted average of the recommendations. This technique is used by Arkin [5] for combining reactive schemas. In the fuzzy inference engine used for behavior fusion, the centroid operation performed on a set of rules does exactly this. The programmer defines the weight of a fuzzy rule by the specifying the set operations in each antecedent. In a single control node, one could think of each rule as being an agent and the centroid operation as a way of reaching agreement between the multiple agents.

3) Fuzzy Multiplexing: As described earlier, the control architecture allows fuzzy rules to recommend values from other nodes. This allows those separate behaviors to be combined at another level according to qualitative rules. Fuzzy multiplexing is a smart version of weighted averaging because the weights for each behavior can change depending on the context of the situation. A component behavior need not yield an appropriate signal at every point of the input space; its effective weight may be decreased toward zero in such situations using the fusion rules.

4) Hierarchical or Supervisory Switching: The subsumption architecture developed by Brooks [1] allows behaviors to shut off or subsume the outputs of other behaviors through the use of a switching operation. Sometimes, an ordered sequence of behaviors is necessary to perform a task. To turn these behaviors on and off, one or more supervisory agents must keep track of the activity. This may be implemented as a finite state machine. The output of a state machine can excite or inhibit the activities of agents within the control environment by using fuzzy multiplexers. Note that a Boolean multiplexer can be implemented using a fuzzy multiplexer by simply making the slope of the membership function vertical.

With these fusion techniques available, the obstacle avoidance behavior can be combined with the goal seeking behavior. The simplest solution would be to add together the steering suggestions from each behavior. However, the magnitudes of the behaviors would have to be carefully adjusted to make sure that obstacle avoidance is strong enough to avoid a collision when goal seeking wants to steer into a wall. A weighted average could be adjusted for this purpose, but the response of each behavior would become subdued. Instead, a fuzzy multiplexer can be used. Intuitively, the obstacle avoidance behavior should dominate the control value when the robot is near obstacles and the goal seeking behavior should dominate when it is not near obstacles. A smooth fusion of both behaviors can be made using the following rules:

```
if closest_sonar is close
then steer_output is avoid_steer
if closest_sonar is not close
then steer_output is goal_steer
```

For an application as complex as a mobile robot, all of these fusion techniques are useful for developing effective reactive behaviors. Fig. 8 shows the interaction between MARGE's low-level navigational agents. Although the individual behaviors are primitive, their combined performance results in a robust and powerful reactive control system. For example, if a goal location exists on the other side of a barrier (as depicted in Fig. 9), the obstacle avoidance and goal seeking behaviors compete and the robot follows the barrier until it finds its way around it. Meanwhile, the barrier following behavior dampens out path oscillations if the surface of the barrier is continuous. Simple concave obstacle arrangements are easily escaped, and dynamic obstacles are dealt with in real time. The ease of integrating new agents into this architecture makes it easy to reconfigure for new tasks and situations. MARGE was entered in the "Office Rearrangement" event of the 1993 Robot Competition hosted by the American Association for Artificial Intelligence. MARGE won first place, wandering among obstacles in search of boxes marked with signs, and moving them into a pattern at one end of the arena [4]. To coordinate this activity, a finite state machine (see Fig. 10) selected goals for the robot (such as signs recognized by the vision system) and set the appropriate context for the behaviors.

D. Object-Oriented Implementation

Software scalability, reusability, and prototyping speed, features of object-oriented design, are useful in the development of complex control behaviors. Although the fuzzy translator and real-time inference kernel used on MARGE was implemented in C, object-oriented languages, such as Smalltalk and C++, can also be used for implementing fuzzy control systems. Smalltalk is easily extensible, allowing fuzzy rules to be defined in-line with the rest of the program code, with no separate translation needed. In a Smalltalk implementation of the translator, classes for fuzzy sets and fuzzy nodes were defined to provide all of the functionality of the translator and kernel implemented in C. Using these classes, fuzzy sets, nodes, and rules were declared in Smalltalk as follows:

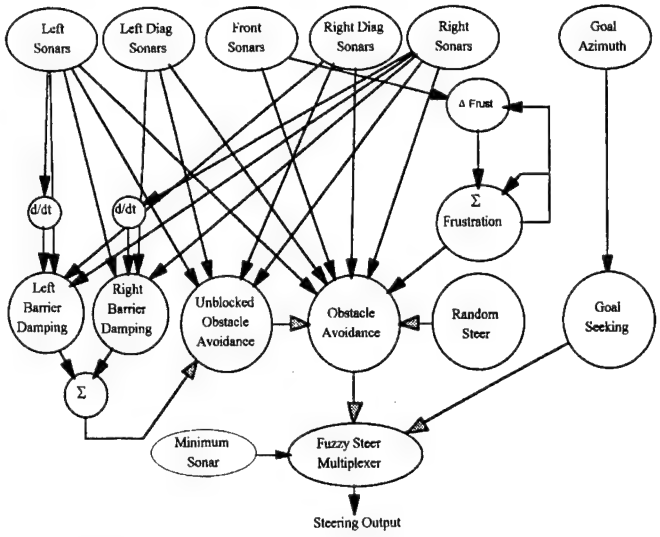


Fig. 8. Fusion of steering agents.

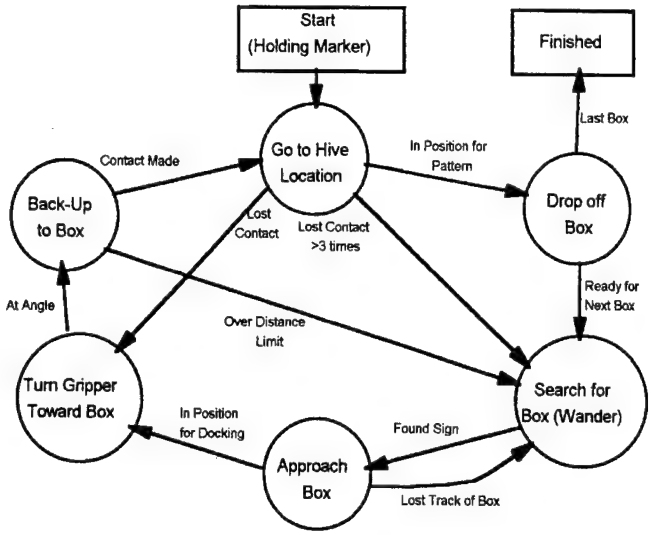


Fig. 10. Behavior sequence for moving boxes.

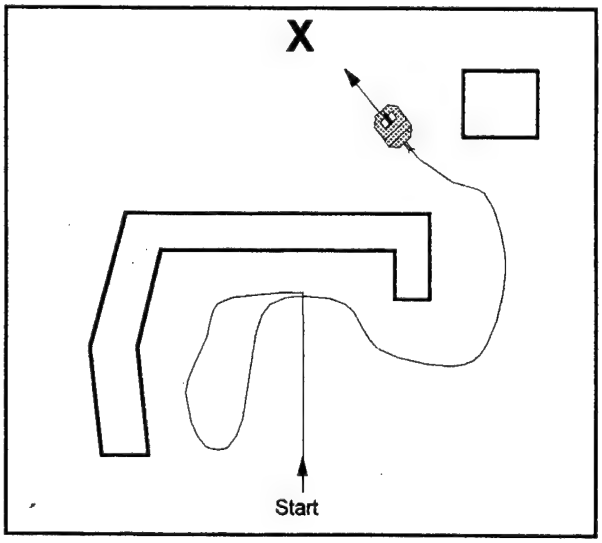


Fig. 9. Emergent behavior of navigation agents.

```
close := FuzzySet high: 100 low: 800.
driveSpeed := FuzzyNode new.
driveSpeed becomes: 0
if: (sonar is: close).
driveSpeed becomes: 100
if: (sonar isNot: close).
output := driveSpeed evaluate.
```

The Smalltalk class Number, which is a superclass of Integers and Floats, was extended to support the `is:` and `isNot:` messages. This allows all numerical values in the language to compare themselves to fuzzy sets. Thus, Smalltalk can provide natural semantics much like the original fuzzy programming language; but Smalltalk's powerful object-oriented roots make it much easier to build upon. Performance of the Smalltalk engine is a little slower than the kernel written in C, but still much faster than needed for mobile robot applications. On a 50-MHz 486 CPU, IBM Smalltalk for OS/2 evaluates an average of over 10 000 fuzzy rules per second.

IV. Conclusion

Fuzzy behaviors provide a convenient abstraction for the development of sensor-based control systems operating in unstructured environments. A large input space can be mapped to actuation by dividing the problem into simpler domains, and developing individual agents that compete and cooperate to perform the task. Fuzzy control rules make the job of defining a control surface easier by providing a linguistic interface to the programmer. Fuzzy behavior fusion has been demonstrated on a mobile robot performing a nontrivial task. Future work will involve the selforganization of behaviors, and the integration of behaviors with high-level planning and mapping systems.

References

[1] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. RA-2, no. 1, pp. 14–23, 1986.
[2] ——, "Intelligence without representation," *Artificial Intell.*, vol. 47, pp. 139–159, 1991.
[3] J. H. Connell, *Minimalist Mobile Robotics*. Boston: Academic, 1990.
[4] I. Nourbakhsh, S. Morse, C. Becker, M. Balabanovic, E. Gat, R. Simmons, S. Goodridge, H. Potlapalli, D. Hinkle, K. Jung, and D. Van Vactor, "The winning robots from the 1993 Robot Competition," *AI Mag.*, vol. 14, no. 4, pp. 51–62, 1993.
[5] R. C. Arkin, "Integrating behavioral, perceptual and world knowledge in reactive navigation," *Robot. Autonomous Syst.*, vol. 6, pp. 105–122, 1990.
[6] L. A. Zadeh, "Fuzzy sets," *Informat. Contr.*, vol. 8, pp. 338–353, 1965.
[7] ——, "Fuzzy logic," *IEEE Comp.*, pp. 83–93, Apr. 1988.
[8] F. G. Pin, H. Watanabe, J. R. Symon, and R. S. Pattay, "Using custom-designed VLSI fuzzy inferencing chips for the autonomous navigation of a mobile robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Raleigh, NC, 1992, pp. 790–795.
[9] G. B. Cunningham, "Integrating fuzzy logic technology into control systems," Tech. Rep., Togai InfraLogic, Inc., Irvine, CA, 1991.
[10] M. Sugeno, T. Murofushi, T. Mŏri, T. Tatematasu, and J. Tanaka, "Fuzzy algorithmic control of a model car by oral instructions," *Fuzzy Sets Syst.*, vol. 32, pp. 207–219, 1989.

[11] K. Konolige, K. Meyers, and A. Saffiotti, "FLAKEY, an autonomous mobile robot," Tech. Rep., Stanford Research Institute International, July 1992.

Steven G. Goodridge received the B.S. degree in electrical engineering from the University of New Hampshire and the M.S. degree in electrical engineering from North Carolina State University. He is currently working toward the Ph.D. degree in electrical engineering.

He is a research assistant in the Department of Electrical and Computer Engineering at North Carolina State University, Raleigh. His research interests include audio and video sensor fusion for camera gaze control, multimedia, and robotics.

Michael G. Kay (S'88–M'90) received the B.A. degree in economics and the M.S. degree in industrial and systems engineering from the University of Florida, Gainesville, and the Ph.D. degree in industrial engineering from North Carolina State University, Raleigh.

He is an Assistant Professor in the Department of Industrial Engineering at North Carolina State University. His current research interests include sensor-based control of free-ranging AGV systems, multisensor integration and fusion, material handling, facilities layout and location, and engineering metrology. He is co-editor of the book *Multisensor Integration and Fusion for Intelligent Systems*.

Dr. Kay has served as Technical Program Co-Chair of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems.

Ren C. Luo (M'82–SM'87–F'92), for a photograph and biography, see p. 345 of this issue of this TRANSACTIONS.

Projection Learning for Self-Organizing Neural Networks

Harsh Potlapalli and Ren C. Luo, *Fellow, IEEE*

***Abstract*—A new learning scheme, called projection learning (PL), for self-organizing neural networks is presented. By iteratively subtracting out the projection of the "winning" neuron onto the null space of the input vector, the neuron is made more similar to the input. By subtracting the projection onto the null space as opposed to making the weight vector directly aligned to the input, we attempt to reduce the bias of the weight vectors. This reduced bias will improve the generalizing abilities of the network. Such a feature is important in problems where the in-class variance is very high, such as, traffic sign recognition problems. Comparisons of PL with standard Kohonen learning indicate that projection learning is faster. Projection learning is implemented on a new self organizing neural network model called the reconfigurable neural network (RNN). The RNN is designed to incorporate new patterns online without retraining the network. The RNN is used to recognize traffic signs for a mobile robot navigation system.**

I. Introduction

NEURAL network models are currently being used in a number of character recognition applications. Commonly used models are back propagation, adaptive resonance models, and self-organization [1].

Of these models, the first two are multilayer networks while self-organization models are single layer networks. The perceptron model is useful for discriminating between clusters with simple class boundaries [2]. The corresponding backpropagation rules were developed by many researchers independently; the work by Rumelhart and McClelland, [3], gives a good background and development of these rules. Hopfield has developed a network for recovering patterns from noisy inputs [4]. Fukushima's neocognitron as well as LeCun's receptive field architecture have been used for translation and scale invariant character recognition applications [5], [6]. Grossberg and Carpenter's ART models are hybrid in that they use both supervised as well as unsupervised learning [7], [8].

On-line adaptation is not possible in multilayer networks due to their architecture. New patterns cannot be learned by the network without retraining with the entire training set. These models have been used for character recognition applications where the extraction of the character is left to other algorithms which also threshold, align, and scale the character. Thresholding algorithms on complex images such as traffic signs could lead to loss of critical information in the sign. Also, once the architecture of the multilayer network is fixed, there is no provision to accomodate an increase in the number of classes without a severe redesign penalty.

Self-organization models offer more flexibility in the design and adaptation of the neural network [9] in terms of reconfigurability to accomodate changes in the number of classes. Amari has discussed a self-organization model for concept formation as well as mathematical foundations for self-organization in [10]. In some cases self-organization models require a smaller number of parameters than a corresponding multilayer network which leads to faster training and recognition times. Self-organization models can have capabilities to adapt to new pattern classes. In this paper we present a new learning algorithm called projection learning. This learning scheme is applied to a self organization model called the reconfigurable neural network (RNN) model. The RNN has the ability to adapt to new input patterns without need for retraining. The RNN is used for traffic sign recognition.

The problem of traffic sign recognition is an interesting one for the following reasons: 1) The within-class variance tends to be high due to changing imaging conditions (light, distance, orientation). It is difficult to cluster classes that are very spread out. 2) The information content of a sign is high. A sign pattern cannot be trivially binarized. Most gray-level variations within a sign carry important information. A simple edge-based model cannot be easily constructed for the sign. 3) Unlike character recognition problems where the numbner of classes is fixed, the number of classes in traffic sign recognition is not fixed. There are a large number of sign patterns that we can observe on the streets. It is prohibitive to store all of these signs in a neural network. We have to compromise by choosing to learn a few of the more important ones. We also require that if a new sign does appear, the network must not break down but should have mechanisms to alert the controller (a master navigation program or a human) and incorporate the new class into the database.

In our model we do not perform any feature extraction on the input traffic sign patterns. This produces considerable savings in computations. Also, by choosing to represent the data from the traffic sign images as continuous values, we are able to retain most of the information from the sign. The additional neurons (represented by weight vectors) in the weight space provide two functions: 1) multiple neurons can be used in classes with high variance, and 2) free neurons can be assigned to new classes. The next sections of the paper describe projection learning and the architecture of the RNN model. We examine optimality issues with regard

Manuscript received March 25, 1995; revised August 9, 1995.

The authors are with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911 USA.

Publisher Item Identifier S 0278-0046(96)02380-5.

to the learning rate. We will present performance curves of the network to show its advantages over standard Kohonen learning. Finally we present our conclusions about this model.

II. Projection Learning

The training algorithm in self-organization attempts to find an optimal cover of the input space by the neurons. Training is concluded when the neuron assigned to each class reaches a predefined level of "similarity" with the inputs. The trained network can be used for pattern classification applications. For each test input vector, the most similar neuron is computed and the input is assigned to the class represented by this "winning" neuron. One of the commonly used similarity measures is the Euclidean distance [11]–[13] with various variations. In this paper we present a new learning scheme based on projections of the neuron onto the input vector and onto the null space of the input vector.

The learning process in self-organization can be best understood if we visualize each neuron as the center of an N-dimensional hypersphere that contains all the examples of the particular class. Initially, the neurons lie at random positions in the input space and they are moved slowly to these centers during the training. If the variance in a class is high, then, as the hypersphere is moved to cover certain members of the class, other members may be left out. If the diameter of the hyperspheres is increased then the spheres from different classes may intersect. This leads to oscillations during the training.

A. Projection Model

The objective is to design a self-organizing feature map with robust learning rules which can be proved to be stable and convergent. We define a convergent series as one in which the magnitude of the terms is steadily approaching zero. In our case, a convergent learning algorithm is one that has a learning error steadily approaching zero. It is not necessary for the error to reach zero. It is sufficient that the error should approach zero. We define stability in the network to be the case when the neurons do not switch classes during training. While there will be some oscillations in the early stages when all the neurons are closely clustered, we want to have a network in which the neurons belonging to each class will only move within the cluster. This will ensure low error rates.

Let the vector $\vec{x}$ denote an input vector and the vector $\vec{w}$ denote a weight vector. The vector $\vec{x}^{\perp}$ will denote a vector normal to $\vec{x}$. Self-organized learning is based on the updates of a "winning" neuron. In the RNN model using projection learning the winner neuron $\vec{w}_k$ is the neuron that maximizes $\langle \vec{w}_k \cdot \vec{x} \rangle$ over all k, where $\langle \cdot \rangle$ denotes the vector dot product. We assume the vectors are all normalized to unit length. By subtracting the projection onto the null space as opposed to making the weight vector directly aligned to the input, we attempt to reduce the bias of the weight vectors. This reduced bias will improve the generalizing abilities of the network. Obviously, the winning neuron is the one that is most closely aligned with the input vector. The update rule is then given by

$$\vec{w}_k(n) = \vec{w}_k(n-1) - \alpha \langle \vec{w}_k \cdot \vec{x}^{\perp} \rangle x^{\perp} \tag{1}$$

where α is the learning rate. Usually these vectors are of high dimension, N. While $\vec{w}$ may be initialized by random values, and the $\vec{x}$ are known, except for when the dimension is 2, there are $N-1$ possibilities for the $\vec{x}^{\perp}$. An effective way to reduce the difficulty in selecting $\vec{x}^{\perp}$ is to first initialize $\vec{x}^{\perp}$ to random values, not all zero. Then compute $\langle \vec{x} \cdot \vec{x}^{\perp} \rangle = \sum_{i=1}^{N} x_i x_i^{\perp}$. If the two vectors are normal to each other then the dot product must be zero. Rewrite the summation as

$$\langle \vec{x} \cdot \vec{x}^{\perp} \rangle = \sum_{i=1}^{N-1} x_i x_i^{\perp} + x_N x_N^{\perp} \tag{2}$$

which reduces to

$$\sum_{i=1}^{N-1} x_i x_i^{\perp} + x_N x_N^{\perp} = 0 \tag{3}$$

or, in other words

$$x_N^{\perp} = -\frac{\sum_{i=1}^{N-1} x_i x_i^{\perp}}{x_N} \tag{4}$$

provided $x_N \neq 0$. Otherwise the first nonzero element of $\vec{x}$ can be used in place of the last element without changing the nature of the equation. The learning rule attempts to increase the projection of $\vec{w}$ along $\vec{x}$ by reducing the projection along $\vec{x}^{\perp}$.

A commonly used weight-update method is based on reducing the Euclidean norm between the weight vector and the input vector. This method has been shown to be successful in many character recognition applications. In applications such as traffic sign recognition, where, the variability within a class may be higher (due to changing imaging conditions), the Euclidean norm tends to bias a weight vector toward a particular input, thereby reducing the generalizing ability of the weight vector.

By using projection learning, we are attempting to reduce the contribution of the entire null space of the input vector class as opposed to making the weight vector similar to any one input vector. Ideally, we would like to compute projections to all the $N-1$ vectors normal to the N-dimensional input vector and subtract out the largest projection as described in (4). This could lead to faster convergence. However, larger memory would be required to store $(N-1)$ N-dimensional normals for each input vector. Further, instead of computing just one projection as described above, we would have to compute $N-1$ projections. Given enough computer resources, a learning strategy which eliminates projections onto the entire null space would yield faster convergence in terms of the number of iterations, although the computation cost will be higher compared to Euclidean learning. In this paper, we have used a compromise approach in the experiments presented here wherein we compute only one normal vector and optimize the weight vector with respect to that normal.

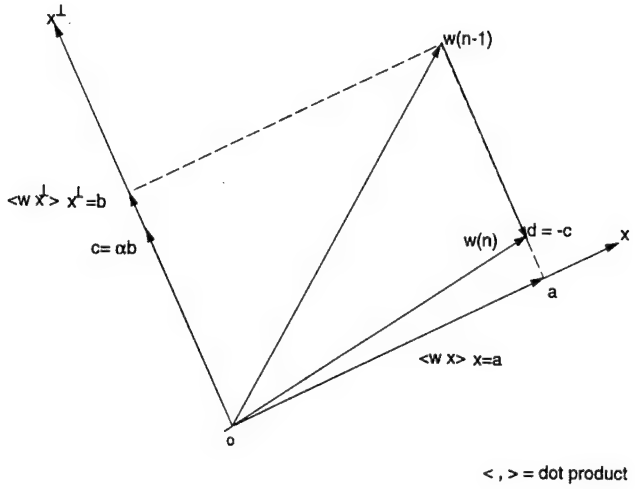


Fig. 1. Example of projection learning.

B. Convergence

We need to show that the learning rule will ultimately yield an update of $\vec{w}$ that will be very close, or identical to, the input. First we prove this geometrically. Fig. 1 is an example of projection theory based learning with learning rate, α.

Here, we see that $\vec{a} + \vec{b} = \vec{w}$, where $\vec{a} = \langle \vec{w} \cdot \vec{x} \rangle$ and $\vec{b} = \langle \vec{w} \cdot \vec{x}^{\perp} \rangle$. The vector used in the learning is $\vec{c} = \alpha \vec{b}$. The updated neuron $\vec{w}(n) = \vec{w} + \vec{d} = \vec{w}(n) - \vec{c}$ does not yield $\vec{a}$ but does create a neuron vector that is much closer to the input than earlier. It is also clear that if the learning rate is greater than one, the adaptation will be unstable and lead to oscillations in the $\vec{w}$. There are two choices for α. We can choose a fixed step size or we can choose a strategy that changes α depending upon the training/test-set errors. The second approach tends to be faster in terms of convergence. For faster convergence we choose a variable α as described below. It is difficult to determine an ideal value for α since it is very problem-dependent. If fast convergence with the possibility of some misclassifications is acceptable, higher values can be used. If zero (or, very few) misclassifications are required a small α should be used.

Since $\vec{w}$ will progressively get more and more aligned with $\vec{x}$, a learning strategy can be developed. Normalization of $\vec{w}(n)$ will eventually yield a neuron that is identical to the input.

Two points should be noted here. If the winning neuron is already normal to the input vector, the $\vec{w}$ will be aligned with $\vec{x}^{\perp}$ and the neuron will be stuck at this point. No adaptation will be sufficient to move the neuron away from this point. In this case, we suggest a delta perturbation of all the elements of the winning neuron to move it away from this trap. The second point to consider is when the angle between the winning neuron and the input is less than 0, that is when the angle is between 90° and 270°. In this case, during the adaptation, the angle between the two will be gradually reduced. At some point the angle must pass from $90 + \delta$ to $90 - \delta$. If the transition does not stop at 90 then the learning will eventually converge. However, if the transition stops at 90 then the local trap phenomenon occurs again and as before, we use delta perturbations to move the vector from the trap. To avoid this problem and to speed up learning, every winning neuron that makes an obtuse angle with the input is reflected through the origin to yield a new winning neuron that now gives a positive dot product.

Now we examine the convergence from a vector product point of view. Let

$$w'_{(k+1)} = w_k - \alpha \langle w_k \cdot x^{\perp} \rangle x^{\perp}$$

and

$$w_{k+1} = \frac{w'_{k+1}}{\|w'_{k+1}\|}$$

For convenience we shall denote $\alpha \langle w_k \cdot x^{\perp} \rangle$ as β. We shall use proof by contradiction. Let us assume that

$$\langle w_{k+1} \cdot x \rangle < \langle w_k \cdot x \rangle \Rightarrow \sum_i^N w_{k+1,i} x_i < \sum_i^N w_{k,i} x_i$$
$$\Rightarrow \sum_i^N (w_{k+1,i} - w_{k,i}) x_i < 0. \quad (5)$$

Now

$$w_{k+1,i} = \frac{w'_{k+1,i}}{\sqrt{\sum_j^N (w'_{k+1,j})^2}} = \frac{w_{k,i} - \beta x_i^{\perp}}{L_{w'}}.$$

Substituting for $w_{k+1,i}$ in (5) we have

$$\sum_i^N \left(\frac{w_{k,i} - \beta x_i^{\perp}}{L_{w'}} - w_{k,i} \right) x_i < 0$$
$$\frac{1}{L_{w'}} \sum_i^N (w_{k,i} - \beta x_i^{\perp} - L_{w'} w_{k,i}) x_i > 0$$
$$\Rightarrow (1 - L_{w'}) \sum_i^N w_{k,i} x_i - \beta \sum_i^N x_i x_i^{\perp} < 0. \quad (6)$$

The second summation is the dot product of a vector and its normal and will go to zero. Also, $L_{w'}$ is a distance that is at least zero and at most 1 (since the vectors are always normalized to unit length). As long as $\vec{w}$ and $\vec{x}$ are not normal to each other, the summation will always be greater than or equal to zero. Therefore, the updated neuron must be closer to the input than the original neuron. In other words, the update will always lead to closer representation of $\vec{x}$ by $\vec{w}$.

C. Learning Rates

Now we must examine if the updated neuron will ever reach an optimal solution. We define optimality as the perfect alignment of the neuron with the input. During the training process, the learning rate is gradually reduced. Let the learning rate at the ith iteration be represented by α_i. Let γ_0 represent $\langle w_0 \cdot x^{\perp} \rangle$, which is the projection of the initial neuron along the normal of the input. Then, following the update sequence

we have

$$\begin{aligned}
w_0 &= w_0 \\
w_1 &= w_0 - \alpha_1\langle w_0 \cdot x^{\perp}\rangle x^{\perp} \\
w_2 &= w_1 - \alpha_2\langle w_1 \cdot x^{\perp}\rangle x^{\perp} \\
&= w_0 - \alpha_1\langle w_0 \cdot x^{\perp}\rangle x^{\perp} \\
&\quad - \alpha_2\big\langle (w_0 - \alpha_1\langle w_0 \cdot x^{\perp}\rangle x^{\perp}) \cdot x^{\perp}\big\rangle x^{\perp} \\
&= w_0 - \alpha_1\gamma_0 x^{\perp} - \alpha_2[\gamma_0 - \alpha_1\gamma_0]x^{\perp} \\
&= w_0 - [\alpha_1\gamma_0 + \alpha_2\gamma_0 - \alpha_2\alpha_1\gamma_0]x^{\perp} \\
w_3 &= w_2 - \alpha_3\langle w_2 \cdot x^{\perp}\rangle x^{\perp} \\
&= w_0 - [\alpha_1\gamma_0 + \alpha_2\gamma_0 - \alpha_2\alpha_1\gamma_0]x^{\perp} \\
&\quad - \alpha_3[\gamma_0 - \alpha_1\gamma_0 - \alpha_2\gamma_0 + \alpha_2\alpha_1\gamma_0]x^{\perp} \\
&= w_0 - [\alpha_1 + \alpha_2 + \alpha_3 - \alpha_2\alpha_1 - \alpha_3\alpha_1 \\
&\qquad - \alpha_3\alpha_2 + \alpha_3\alpha_2\alpha_1]\gamma_0 x^{\perp}.
\end{aligned}$$

Following this pattern, we can write the general equation for the nth update of the neuron as

$$w_n = w_0 - \gamma_0\left[\sum_{i=1}^{N}\alpha_i + \prod_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N}\alpha_i\alpha_j\right]x^{\perp}. \tag{7}$$

Ideally (7) should converge to the standard form as shown in (1) with learning rate (also Fig. 1). The learning rate is a monotonically decreasing function. When the learning rate goes to zero, one of the product terms in (7) vanishes. The key term is the sum of the learning rates. From [11], one of the conditions for convergence is

$$\alpha(n+1) \geq \frac{\alpha(n)}{1+\alpha(n)}. \tag{8}$$

This also ensures that the estimate, w_n, of the centroid of the input set has zero variance. We use a linear decay function (as opposed to an exponential decay) for the learning rate since, as shown by [11], such as schedule favors more recent input presentations. This makes the network more sensitive to recent changes in the input patterns. This is useful when we expect to see some new patterns during later stages of the training/testing. Exponential schedules favor data presented at the beginning of the training. The convergence of instantaneous gradient learning schemes such as the one shown above has not been proved yet since, among other factors, it is difficult to measure the change in each Voronoi region after each adaptation. Also, the reduction in the distortion measure is not always monotonic. However, in practice convergence has always been obtained with suitable choice of parameters.

At the terminal stage of the training, we have the standard form of the update equation as

$$w_n = w_0 - \gamma_0 x^{\perp} \tag{9}$$

which is the neuron with the projection along the normal to the input removed, that is, the updated neuron vector lies entirely along the input vector. Thus, the solution will eventually converge to an optimal solution provided that the summation tends to unity.

D. Neighborhood Update

A small monotonically decreasing neighborhood of the neurons around the winning neuron is also updated with respect to the input. The idea behind this approach being that neurons that are close together typically belong to classes that are close together. For example, let at time 0 (before training the network) neuron k be the closest neuron to neuron l. Also, let sign class K be most similar among all the classes to class L. Finally, let neuron k be assigned to class K and let neuron l be assigned to class L after training. During training, when a member sign of a class is presented to the network, the closest neuron is updated. Now, if we do not use neighborhood update in this example, neurons k and l would be individually updated in separate passes even though, eventually, they will end up in the same part of the hyperspace. However, if we use neighborhood update (as proposed by Kohonen) every time neuron k is update, neuron l is also moved in that direction by a small amount and vice versa. Thus, neighborhood update improves learning rates.

III. Reconfigurable Neural Networks

Reconfigurability is invoked when new objects that were not seen during the training phase appear during the testing phase. In this case, the objective is to incorporate the new traffic sign patterns into the network memory. The detection and integration of new classes will be presented in this section.

During the training phase each neuron keeps track of the farthest member of its class. In the sense of projections, the farthest member is the one that has the smallest dot product with the representative neuron of its class. This is called the *activation threshold*. The dot product between this neuron and members of other classes will be smaller than the threshold. We start the network with an excess of neurons, that is the number of neurons available is greater than the expected number of classes.

The on-line network also maintains an activity counter for each assigned neuron. Whenever the neuron is the "winner" the counter for this neuron is incremented. When a new class is presented to the network, the dot product between the input and all assigned neurons is computed and the neuron is assigned to the class where it has the largest dot product. If, however, this value is smaller than the activation threshold of this neuron, then a new class has been detected. Free neurons are then moved to the location of the new class. If there are no free neurons available, then the neuron with the lowest activity counter is moved to represent the new class. This is best illustrated by the example shown in Fig. 2. The outer circle represents the weight space. The black circles represent new classes. Initially, the new inputs are classified as members of the classes near the bottom of the outer circle. However, it is clear that the new inputs are much farther than any other member of the class. Therefore, they are detected as new classes and one of the free neurons is moved to this location.

The first appearance of a sign pattern of a new class is easily detected by the RNN model. The problem that arises when subsequent sign patterns of this new class, or other new classes are applied to the RNN is a problem of resolution. How can we

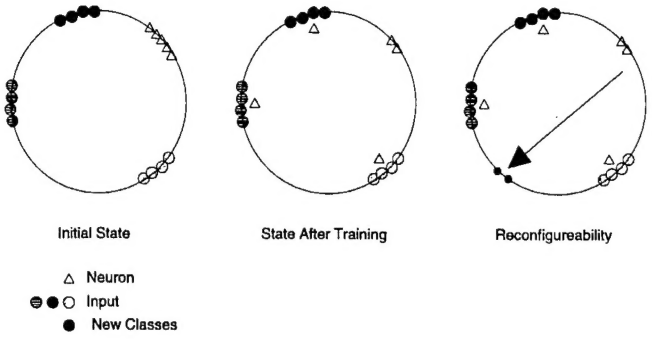


Fig. 2. Detection and incorporation of new classes: The circle on the left shows the initial state of the weight space; there are three well-defined clusters and five neurons. The circle in the middle shows the state after training. Each class is represented by one neuron and there are two free neurons. The circle on the right depicts the unique feature of the RNN. When patterns from a new class appear (represented by black circles), first either of the closer neurons will fire. However, since the new patterns are outside the activation threshold of the old classes, the new patterns will be classified as a new class. One of the free neurons will be assigned to this new class. If no free neurons are available, one of existing classes with the least activity will be discarded and the freed neuron will be assigned to the new class.

decide if the subsequent sign pattern belongs to the new class or if it is yet another new class? This problem can be resolved by setting the distance between the neuron of the new class and its first member as the activation threshold for the new class. If the subsequent sign pattern is not more than the activation threshold $+\delta$ distance away, then it belongs to the same new class. Else, it is another new class. If it belongs to the same new class, then the activation threshold is also updated. In any case, the major advantage of the RNN is its ability to add the new patterns without retraining. The RNN can be set up to flag when new classes are detected. An operator (human or higher-level controller) can then decide the characteristics of the new pattern, its relative importance to the overall strategy (which may be navigation, target recognition, etc.) and then choose to retain the new pattern (and assign a label to it) or discard it.

The number of additional neurons that we wish to include in the neural network is problem dependant. Ideally, we should be able to predict the number of classes in advance. In most character recognition applications, the number of classes (i.e., characters) is fixed. However, in the case of traffic signs, a new sign (or, target) may appear. In our experiments, all unassigned neurons are tagged as free neurons that can be assigned later to a new class. Typically, 10% of extra neurons should be enough to capture new classes without retraining. If the network sees too many new classes this indicates a design deficiency. Also, even if all the free neurons are used up, neurons that belong to a class that has been dormant for a long time (based on a "hit-count" maintained in the network memory) can be freed up to accomodate new classes.

IV. Experimental Results

The network is trained with images of traffic signs extracted from video images captured by the vision system on the MARGE mobile robot at our laboratory [14]. Some examples are shown in Fig. 3. The smallest size of the sign was 20×20 pixels where the sign was barely visible to the human eye. The largest size was 60×60 pixels when the camera was nearly alongside the sign.

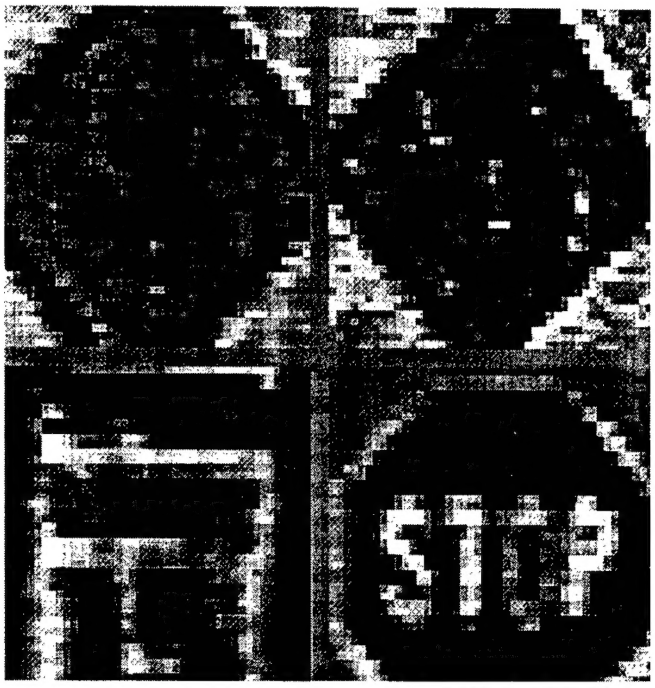


Fig. 3. Examples of traffic signs. Clockwise from top left: Do-not-enter, left-turn, stop, speed-limit.

TABLE I
Projection Learning Parameters

Number of signs	Examples per sign	Training	Testing	Number of Neurons (total)
10	100	70	30	40

A. Comparisons

A bounding box of size 45×45 is placed with its center at the center of the region identified as a sign. Only the region within this box is passed to the neural network. Here we are knowingly introducing distortions into the patterns. When the actual size is smaller, clutter from the neighboring regions is added to the landmark image. When the actual size is larger, some information from the traffic sign is lost. This distorted training set increases the generalizing ability of the network.

The gray levels in the region are scaled linearly between -1 and 1 with gray level 0 set to -1 and gray level 255 set to $+1$. We can expand the resolution of a particular sign by scaling the gray levels of that sign alone from -1 to $+1$. This method has the apparent disadvantage that a different sign with same range of gray level values would also scale to the same floating point values. However, the distribution of these floating-point values would be different for different signs. Hence, using the range of gray-levels of each sign as a scale for that sign-class would yeild a high resolution conversion.

For each sign type extracted from the video image, 100 examples were created with equal representation of each example of that sign. Uniform random noise of upto 10% was added to all the examples. Examples that were clean were also occluded by placing a 4×4 box at random locations on the sign pattern. The signs were divided in a $70 : 30$ proportion between training set and test set patterns. The following table lists the network parameters.

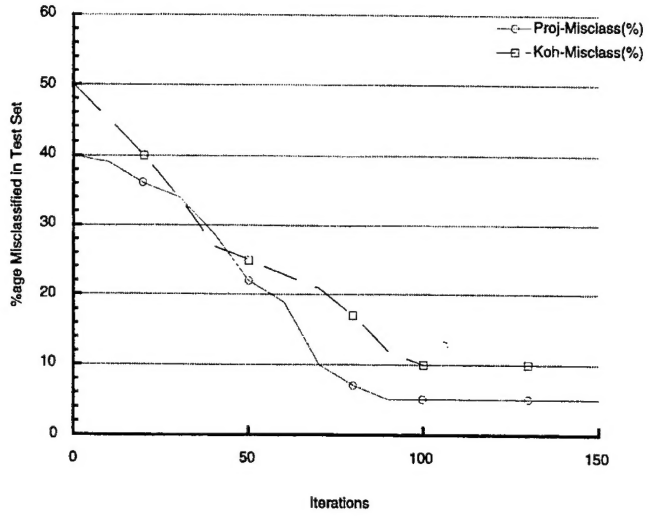


Fig. 4. Comparison of misclassification errors: Kohonen learning versus projection learning.

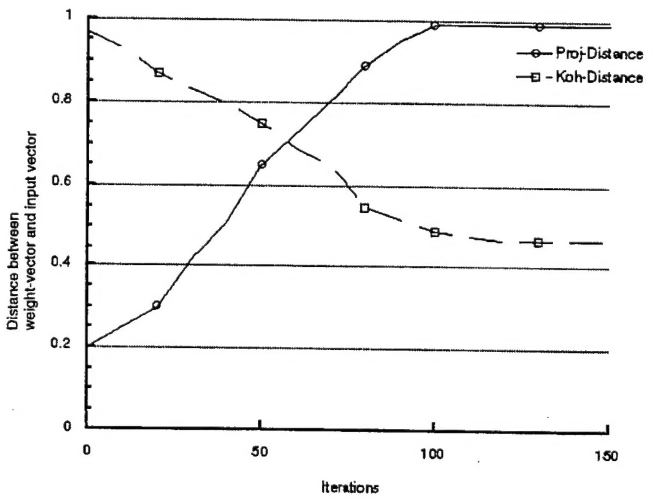


Fig. 5. Comparison of input-neuron similarity: Kohonen learning versus projection learning. In Kohonen learning distance between weight vector and input decreases over time. In projection learning dot product increases over time.

The test set patterns were presented periodically during training to gauge the learning. No adaptation was allowed in the testing phase. Fig. 4 shows the misclassification errors on the test set for standard Kohonen learning and projection learning. These curves show results averaged over 100 trials. Learning based on null-space projection shows distinct advantages.

Fig. 5 shows the mean distance between the winning neuron and the input for Kohonen learning and the mean dot product between the same for projection learning. The distance measurement taken over 45×45 dimensional space fails to give a good picture of how well the neurons represent the input. On the other hand, we can see that in the case of the dot product representation, the angle gives a measure of colinearity which can be used to measure similarity. For example, a dot product result close to unity indicating an angle close to zero would imply high similarity.

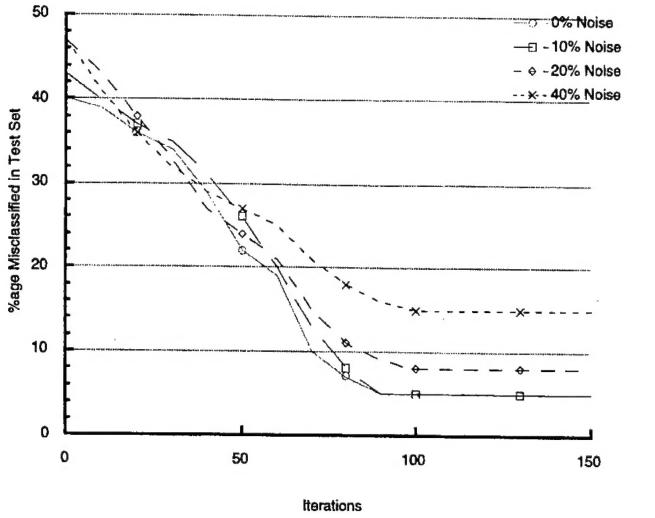


Fig. 6. Effect of noise on learning rates.

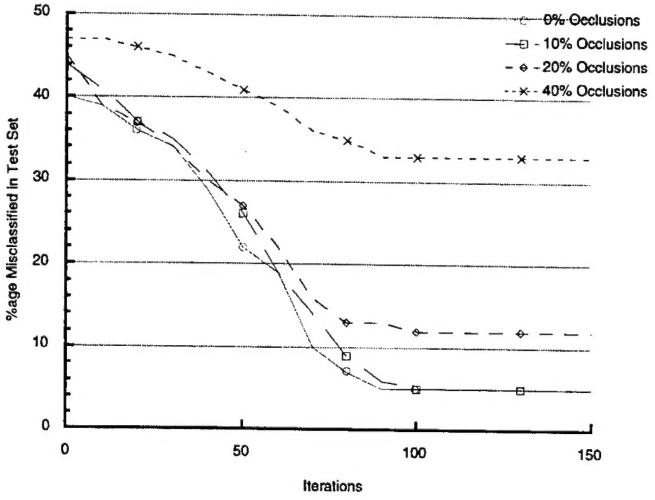


Fig. 7. Effect of occlusions on learning rates.

B. Performance Issues

Several training sets were generated from the original set by adding noise to varying degrees. Fig. 6 shows the effect of noise on the learning rates. The numbers in percentage for each curve indicate the maximum noise level for that training set. For example, noise level of 20% indicates that the intensity at each pixel was randomized by upto $\pm 20\%$. As we might expect, the learning is degraded for severe noise conditions. However, for moderate noise, the network is able to learn the signs.

In another experiment, parts of the signs were occluded by randomly placed rectangular windows. Several training sets were generated each with varying sized windows: 5×5, 10×10 and 20×20. In the last case, upto a quarter of the sign was occluded. Fig. 7 shows the performance of the network when the traffic signs were occluded. The network is able to learn most of the occluded signs.

C. Reconfigurability

The free neurons in the network serve two functions. First, the free neurons are also available to assign to new classes that

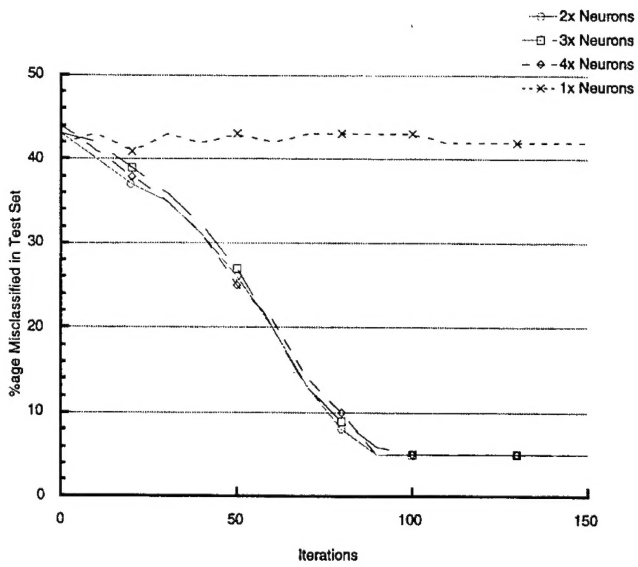


Fig. 8. Effect of excess neurons on learning.

may appear at a later time. Second, if the variance within any class is high then, one neuron may not be able to represent the entire class. In this case, we can allocate multiple neurons to the class. This class will be assumed for an input when any of the multiple neurons is the winner. However, there must be an upper limit to the number of neurons needed for any class. We must remember that the search time for the best match is proportional to the number of assigned neurons in the network. Therefore, there is a trade-off between the amount of neurons needed and the maximum search time that can be allowed. We trained the network with noisy data and increased the number of neurons during each trial. Fig. 8 shows the different learning curves. We observe that there is a minimum number of neurons below which the interclass variance is so high that the network is unable to learn at all. We also see that there is an upper limit where additional neurons do not affect the learning rate. This is because each class must be a cluster of relatively closely spaced inputs. Once the sufficient number of neurons move into the cluster, no other neuron can enter the cluster.

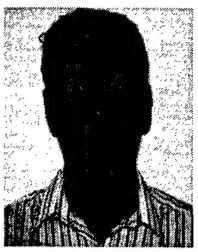
The trivial case of assigning one neuron to every input is not presented in these curves. In this case the learning would cease at the end of the first iteration. However, the dimensionality reduction that we expect from the learning process would be absent making the network useless.

V. Conclusion

In this paper a new learning method for self-organizing neural networks is presented. A new neural network architecture capable of incremental learning is discussed. This network learns the inputs by subtracting from the neuron its projection onto the null space of the input. By subtracting the projection onto the null space as opposed to making the weight vector directly aligned to the input, we attempt to reduce the bias of the weight vectors. This reduced bias will improve the generalizing abilities of the network. The optimality conditions for this network are also presented. The learning curves of the network are presented and it is shown that the network is able to recognize the traffic signs that are in the database successfully. We have described how reconfigurability is implemented and how it is used to learn new sign patterns. While the network has the ability to learn and recall sign patterns, it does not have knowledge of the information content of a sign. This is, in fact, true of all neural network models: the networks can only detect which class the input belongs to. Our model has the added ability to detect new classes and to add them into the database as well. What action to take when a particular pattern is detected, or if a new pattern is detected, is not the job of this network, or in general of any network; that task belongs to the controller program that invokes the neural network for the recognition task.

References

[1] J. Zurada, *Introduction to Artificial Neural Systems* New York: West, 1992.
[2] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychology Rev.*, vol. 65, pp. 386–408, 1958.
[3] J. Rumelhart and J. McClelland, *Parallel Distributed Processing.* Cambridge, MA: MIT Press, 1986.
[4] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Academy of Sciences*, vol. 79, pp. 2554–2558, 1982.
[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, 1989.
[6] K. Fukushima, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. 13, pp. 826–834, Sept./Oct. 1983.
[7] G. Carpenter and S. Grossberg, "Art2: Self organizing of stable category recognization codes for analog input patterns," *Applied Optics*, vol. 26, no. 23, pp. 4919–4930, 1987.
[8] ——, "A massively parallel architecture for self organizing neural pattern machine," *Computer Vision, Graphics and Image Processing*, vol. 37, pp. 54–115, 1987.
[9] T. Kohonen, *Self Organization and Associative Memory.* New York: Springer-Verlag, 1988.
[10] S. Amari, "Mathematical foundations of neurocomputing," *Proc. IEEE*, vol. 78, pp. 1442–1462, Sept. 1990.
[11] E. Yair, K. Zeger, and A. Gersho, "Competitive learning and soft competition for vector quantizer design," *IEEE Trans. Signal Processing*, vol. 40, pp. 294–309, Feb. 1992.
[12] P.-C. Chang and R. Gray, "Gradient algorithms for designing predictive vector quantizers," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-34, pp. 679–690, Aug. 1986.
[13] L. Wu and F. Fallside, "On the design of connectionist vector quantizers," *Computer Speech and Language*, vol. 5, pp. 207–229, 1991.
[14] R. C. Luo, H. Potlapalli, C. Aras, and M. Lin, "Marge: Mobile autonomous robot for guidance experiments," in *4th World Conf. Robotics Research*, Pittsburgh, PA, 1991.



Harsh Potlapalli received the B.E. degree in electrical engineering from Osmania University, Hyderabad, India, in 1987. He received the M.S. degree from Tulane University, New Orleans, LA, in 1989, and the Ph.D. degree from North Carolina State University, Raleigh, in 1994, also in electrical engineering.

He is currently working as a Postdoctoral Research Associate at NCSU. His research interests are in object recognition, texture analysis, autonomous inspection systems, and hybrid vision and neural network systems for unstructured scene understanding.

Dr. Potlapalli is a member of Eta Kappa Nu.

Ren C. Luo (M'82–SM'87–F'92), for a photograph and biography, see p. 345 of the June 1996 issue of this TRANSACTIONS.